HOBBY COMPUTER HANDBOOK

$1·50*
$1.70 N.Z.

SPACE see page 30

BEGINNER'S GUIDE TO COMPUTERS

• How Computers Work • How to Choose a Home Computer • Learn to Speak BASIC Computerese • Learn to Count Like a Computer • Some Hard Facts About Software

COMPUTERIST'S GUIDE TO HARDWARE

• Heath H8 • SWTP 6800 • Radio Shack TRS-80 • Apple II • RCA COSMAC • Pennywhistle Modem • TI-58 • Mainframes • Monitors • Printers • Memories • and Much Much More

COMPUTERS TODAY AND TOMORROW

• Computer Chess Playing • Computers Phone the Future • Computer Space Games • Computers in Your Car • Computer Supermarkets • Computer Communications • Computer Fun

The Apple II is a self-contained computer with an integer BASIC in ROM, which can be used to load a 16K BASIC. It includes at least 4K of RAM, and game paddles, and will output full-color to a full-color video monitor or TV set (with RF modulator).

Q–*What is a "Monitor"?*

A–See **Firmware** in the previous question.

Q–*What is meant by "Boot" and "Boot strap"?*

A–You have probably heard of the expression "Picking yourself up by your bootstraps."; meaning, getting yourself started by moving yourself. Same thing with computers. A computer can only sit there and do nothing until programmed; but you can't just shove a program into the computer; something must tell the computer what's going on when you start to enter your program. A boostrap program is a very small program that sets up the computer to load a larger, complex program. To boot a program means to use a program (generally in a resident monitor or operating system) to program a larger program, or to set up something like a disk operating system.

Q–*Ay accessory I/O I'm planning on adding to my home computer has a feature called "handshaking." Exactly what is a computer's handshake?*

A–**Handshaking** is electronic confirmation that some piece of computer equipment is ready to execute operations. For example, before a computer transmits to a mass storage device such as a recorder it might send out an electronic signal to find out if the recorder is ready. On receipt of the signal the recorder, if ready, will transmit a signal back to the computer that it is ready. The computer will then transmit the start signal, followed by the data transmission. If the computer does not get its "handshake" from the recorder it doesn't transmit data. Handshaking can also work the other way: The recorder might send out its handshake signal, and automatically feed data to the computer only if a handshake is received from the computer.

Modems also generally use handshaking (see next question).

Q–*What's a modem?*

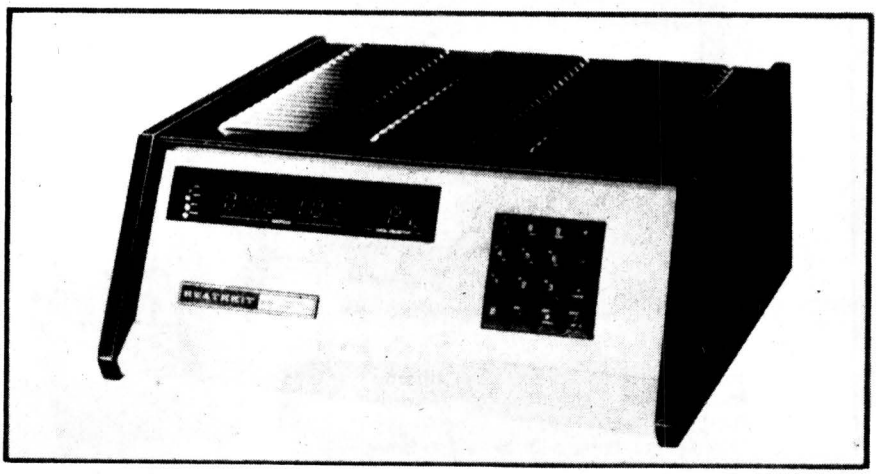
A–A **modem**–a term derived from **mod**ulator/**dem**odulator–is a device that connects both computers and terminals through telephone or other remote-wired circuits. A modem converts the electrical impulses from terminals and computers into audio tones which can be easily handled by voice-grade telephone circuits. Some modems, used on private lines, work at extremely high speeds; the common voice-grade modems operate at TTY speed (110 words per minute) or 300 words per minute. Modems used at terminals are called *originate* modems. Those used at the computer are *answer* modems. Each responds to different audio tones: the originate modem transmits low tones and receives high tones from the computer. The answer modems trans- high tones and receives low tones.

Q–*A group from school would like to set up a computer we could use from each home. Is this possible?*

A–Yes. You will need an answer modem at the computer, and each of you will need a terminal with an originate modem. Some means must be provided, if there is no one to attend the computer, to automatically answer the phone at the computer, connect the modem, and then "hang up" when the terminal signs off. This is easily accomplished through a modems handshaking signal. (We hope to have a construction project on such a device in an upcoming issue.) Answer modems, and combination originate/answer modems, manufactured by Omnitec–perhaps the most respected name in modems–are available from some surplus dealers from time to time. You have to keep looking because answer modems don't come cheap–even used.

Q–*Is there some reason paper tape (teletype) recordings made on my own computer cannot be fed into my school's computer, or the time-share system available through my school?*

A–Yes. Even though computers might use the same CPU there are minute variations in the encoded signals. It is more common than not that recordings from one type of computer cannot be fed to another, even when using the same type of recording system. For example, one of the most



Based on the 8080A, the Heathkit H8 has a 16-key front panel which, in octal numbers, allows you to address registers and memory. I/O interfaces and memory are options, as are peripherals.

popular personal computers uses four nulls and a control signal at the end of each line of a BASIC program; this encoding precludes its acceptances by other personal computers. Similarly, it cannot accept recordings from a time-share system. It's a common problem. Some day there might evolve a common recording or encoding standard.

Q—*What is an "acoustic coupler"?*

A—An acoustic coupler is a modem that is connected to the telephone circuit by placing a telephone's handset in the modem's sound-absorbing cradle —which contains a speaker and microphone to couple sounds from the modem to the phone and vice-versa. This contrasts with a "hard wired" modem that is wired directly to the telephone line(s).

Q—*What would cause intermittent recordings from my computer? I'm using a Kansas City interface and a Panasonic cassette recorder. Sometimes I load a program and find there are errors, even on the safety dump.*

A—If you can get good recordings occasionally, it's a sure sign both the interface and recorder are okay. Most likely you are using a really cheap tape, and dropouts—which normally go unnoticed with sound recordings—are dropping bits out of your dumps. Even at 300 baud you need decent tape such as TDK-AD, Maxell UD, and AVDEX, all of which are excellent to at least 4800 baud. When recording baud rates above 4800 a special data cassette is recommended.

Q—*What is the difference between audio and data cassettes, and why are data cassettes at least twice the cost of audio cassettes?*

A—The primary difference between an audio and data cassette is the pressure pad. The one on the data cassette is oversize, generally made of a special low-friction material, and often costs more than the tape itself. In addition, the tape (supposedly) has a more uniform coating, is less prone to oxide flaking, and most important, is certified for a specific minimal baud rate. It is claimed the shell and internal construction are better but we haven't noticed construction having any effect when it comes to personal computers.

One advantage of the personal computing data cassettes such as those from AVDEX is that they have short tape loads; you don't pay for 30 minutes worth of tape when you need about thirty seconds worth. (Data cassettes often come in several "short" lengths.)

Q—*What is meant by "serial" and "parallel"?*

A—In **serial,** each bit of the seven bits (or eight with parity) making up an ASCII character, or binary information, is transmitted to or from a computer in sequence, one bit after the other. Special timing and encoding tells the computer which bits make up a character. In **parallel** form all bits are simultaneously transmitted, so no timing or special encoding is required. TTY uses a serial format. An inexpensive tape reader such as the Oliver uses the parallel format, thereby allowing you to feed the tape as fast as you can pull it through the reader. All you must be certain of is that the computer's I/O matches the terminal or peripheral: serial for serial and parallel for parallel. You cannot mix the two, such as feeding a serial TTY through a parallel I/O port. (Note. Though a TTY feeds and receives in serial unless specially modified, a TTY punched tape is recorded parallel—the TTY makes the "conversion".)

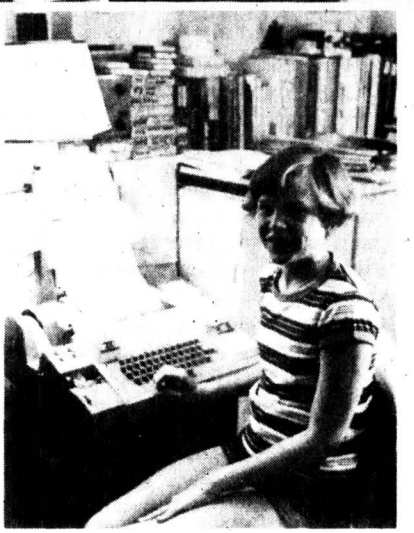
Q—*How much memory can be installed in a personal computer?*

A—The maximum amount of memory is determined by the particular CPU and the size of the power supply. Some kits provide for something like 16K, 20K, or 24K in the main cabinet, with an extra cabinet and power supply needed for additional memory. Other kits have a heavy-duty power supply, usually a cooling fan, and can accommodate up to 48K of memory. New memory ICs draw relatively little current and you can now get 8K of memory in less space than you needed for 4K, and the 8K takes less current. As a general rule, 6K to 8K of available memory—in addition to the memory needed for your higher language such as BASIC —is more than enough for 99% of the average personal computer's programs. You need a lot more memory —upwards of 20K—when you start getting into filing systems, or FORTRAN. But if you're into files and/or FORTRAN you really need a disc system.

Q—*What is a "Header"?*

A—A **header** can be several things, but it generally refers to a code, often a single letter, placed in front of a program when several programs are recorded on the same tape. The computer can be programmed to search for the header and load only the program that follows the specified header. Basically, it's a simplified filing system for cassette storage. Some BASICs make provisions for headers, or more commonly, "files"; others don't, and the BASIC loads everything coming off the tape. ■

This computer hobbyist, Jay Moss of California, started off with a basic Altair microcomputer. He demonstrates how one can expand a system well beyond its original intentions. With a modified-Selectric printer, floppy discs and more he soon discovered the computer's capitalistic tendencies after using it in his plumbing contracting business.



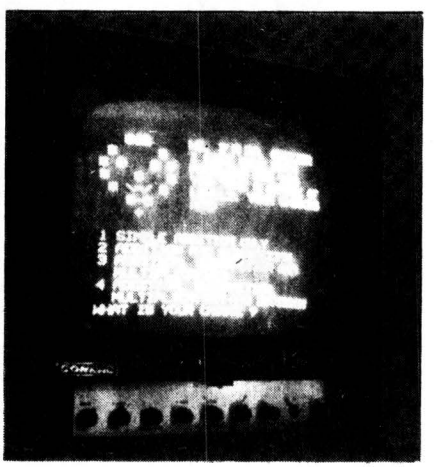
Claudia's Computers

Twelve-year-old shows that age is no barrier to computer fever.

by Charlene Knadle, WB2HJD

☐ Who's using computers these days? Not just adults in business, science and math, if that's who you thought. To prove it meet 12-year-old Claudia Napfel. This Long Island, New York junior high school student is a whiz at operating both her own and her father's computers. But Claudia doesn't stop there—she programs both computers to do math and science problems, to play games and most important to her, to help her enjoy her music.

Claudia says that the computers help her with her two favorite school subjects, science and math. Not surprisingly, these are Claudia's best subjects in school. So that she could use her father's more powerful computer Claudia learned how to program it on his ASR-33 teletype input.

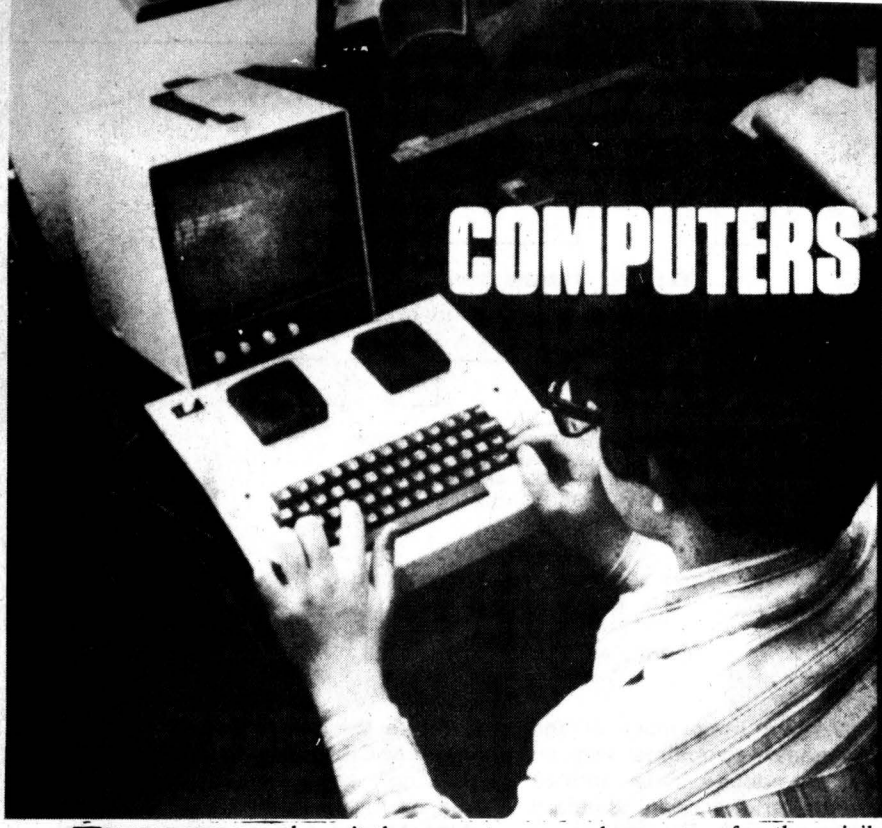


Claudia's computer can play math games and it will quiz her on many levels.

Music Machine. In addition to playing games for fun, Claudia gets a great deal of pleasure out of her music programs. She often writes music and programs the computer to play it back on a synthesizer to see if there are any unusual notes. Says Claudia, "when I write songs the computer helps me with notation and time," and she finds it improves her ability to read notes too.

What do her friends think?

"They call me brainy," said Claudia, "walking dictionary and things like that. But they're just kidding—when they're here they're fascinated." When asked if using a computer ever becomes "old hat," Claudia answered: "No, I'm never bored with it even though it's familiar. I'm fascinated every time I sit down at the terminal." ■



COMPUTERS PHONE THE FUTURE

How to link your home to today's new, computerized nerve-centers.

by Herb and Lawrence Friedman

EVEN IF YOU NOW haven't the vaguest interest in personal computers, within a year or two you will likely be up to your neck in personal computing whether you like it or not; and by *personal computers* we don't mean a Hewlett-Packard or TI programmable calculator—though they are, in fact, personal computers.

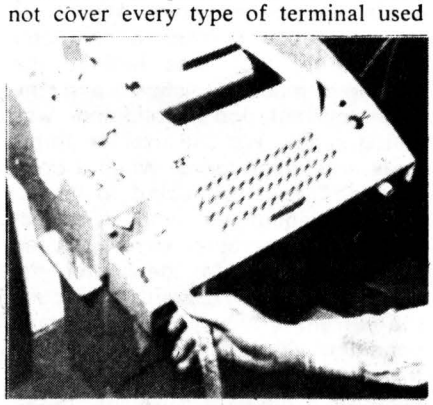
Personal computer means a full blown computer you can use to keep and process business and family records; one your children can, and probably will, use for their math, science and social studies—at all levels from grade school up; a computer your preschool children might use to play games such as *Star Trek* and *Klingon Capture;* a computer you could use for adult education courses taken in the comfort of your own home.

For less than the cost of a decent hi-fi system, or a console color TV with remote control, you can have a computer in your own home as powerful as some of the big IBM jobbies, and the whole computer won't take up much more space than a couple of shoeboxes. If you don't have room for two shoeboxes worth of electronic hardware you can rent computer time from regional and national companies. For example, did you ever wish you had an hour or so of computer time to finish a school problem? *Call Data* will charge you as low as $6 per hour (educational rate) for using their computer. If all you need is a couple of hours on their computer per month the bill could be less than your phone company charges you for the privilege of having a telephone.

Whether you opt for a personal computer in your own home, or a connection through your telephone to a time-share computer service you are going to need a *terminal,* the device that lets you communicate with the computer.

Trouble is, the surplus market is loaded with *computer terminals* that are really worthless for general use, so this article is going to serve as both an introduction to computer terminals and a guide as to what's available and really worthwhile.

Because of space limitations we cannot cover every type of terminal used to communicate with computers. Since the average reader will have access to a time-share dial-up computer, (via the telephone) or a personal computer in the home, we are going to cover only the terminals that work with both systems. There is no reason to spend several hundred dollars for a terminal that works with a personal computer only to find that if someone offers you free time on a giant time-share system your terminal can't be used



By simply feeding punched tape through the reader, as shown, a computer will accept a program at the maximum TTY speed of 110 words per minute. This particular program will require a full thirty minutes of computer time and will check every single bit of memory in an 8K word memory board.

Two Piece Terminals. At the very minimum a computer terminal consists of two separate devices: a typewriter style keyboard that transmits information to the computer, and a display device that "prints" the information coming from the computer. The display can either be some form of printer that "types" a written record—called "hard copy"—on paper; or a CRT, which can be a TV monitor, a modified TV receiver, or anything else using a CRT.

As a general rule the two sections—keyboard and printer—are totally independent even though you might have used, or observed, a working computer and seen that whatever was typed on the keyboard was printed out on the display. The keyboard transmits information to the computer, which then *echoes* back to the display so the user can check the entry. If the user types the word *ready* the computer echoes back the word and the display shows the word *ready*. Without the echo the user would have no idea whether he was transmitting correct data into the computer. The condition of separate

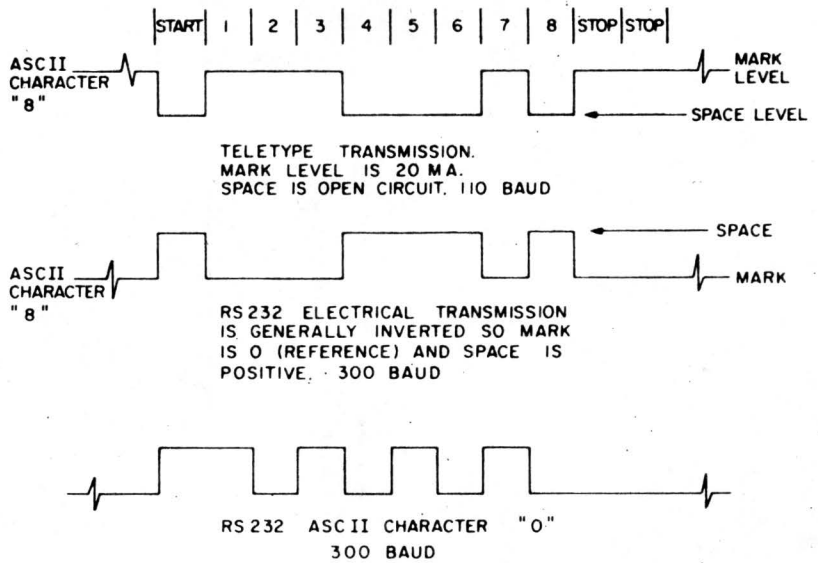
keyboard and display is termed *full-duplex*, meaning two things can occur at the same time—you can transmit and receive simultaneously.

Another condition is termed *half-duplex*, and there appears to be several versions of what is meant by half-duplex. The most common use feeds the keyboard character to the display and the computer. When the computer echoes back, the display shows both the original keyboard entry and the echo. In full-duplex, a keyboard entry of ABCD is displayed via the echo as ABCD. In half-duplex the display shows AABB CCDD. In full-duplex the word *ready* is displayed as READY; in half-duplex it appears as RREEAADDYY Half-duplex transmission is commonly used when time cannot be allowed for the computer to echo back. There are certain conditions when this is necessary, and the half-duplex mode, at the least, allows the user to check *spelling* and codes as they are entered into the computer.

Local is a term meaning the keyboard is connected directly to the printer so that keyboard entries are fed only to the display and any recording device attached to the display. In this manner a tape can be prepared for transmission to the computer at a later time.

If the terminal is in the vicinity of a computer it can be connected directly to the computer via multiple wires—this is termed *hard wired*. On the other hand, it is possible to use the telephone to "converse" with a computer on the other end of town, or even the country. The terminal is connected to a device called a *modem*, an acronym for *Mod*ulator-*Dem*odulator. Some modems can be hard wired directly to the telephone line; others can be acoustically coupled by simply placing the telephone's handset into a built-in receptacle. The modem takes the signal from the keyboard and converts it into audio tones which are fed to the computer through the telephone line. At the receiving end, a different type of modem converts the audio tones back to an electrical signal for the computer. Each modem is a two-way device. The modem at the computer takes the computer's output electrical signal and converts it into audio tones for transmission to the terminal. At the terminal the modem converts the audio tones back into electrical signals needed to drive (produce) the display. To avoid a mix-up on the telephone line the audio tones from the

TECHNIQUES OF ASCII INTERFACING



All time-share and most of the larger personal computers use an electrical character code termed ASCII, short for *A*merican *S*tandard for *C*ommunications *I*nterchange of *I*nformation, and pronounced "as-key."

The 8-level ASCII code presently in use was designed for teletype so the associated terminology refers to TTY operation even if the entire communications system is electronic with a CRT for the display.

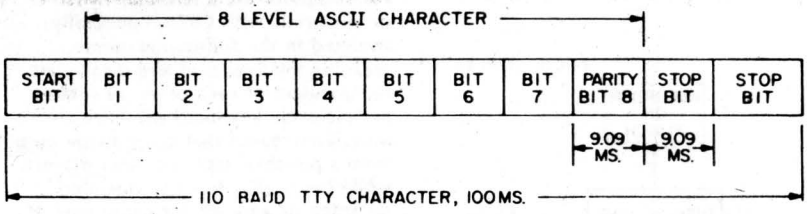
The TTY terminology in turn goes back to the days of the telegraph when one or a series of magnetic sounders was connected in a battery powered loop through the telegraph keys. Since everything was in series, the normal *off* condition was for current to flow in the loop and this condition was called *mark*. When the key was open to send a character the current loop was broken, the sounders made a "click" and the open current loop condition was called *space*. We still use the terms *mark* and *space* today. When a computer's TTY is connected to its associated equipment and it is not receiving a character signal it is fed 20 mA of current for the printer—the *mark* condition. To send characters the current is interrupted—the *space* condition. (You might have to read this over a few times to get the sense because it appears to be the opposite of what's needed. As we'll show, in electronic systems we invert the signal so *mark* is generally zero current or voltage and *space* is some amount of current or voltage.)

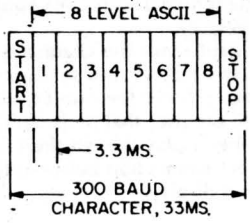
On to the next confusion. The 8-level ASCII character code really uses only seven *bits* to transmit the character; the eighth bit is for something termed *parity*, and even here we don't stop because there must be signals to synchronize the sending and receiving equipment. After all, something has to tell the printer that the electrical pulses to follow are a character and not line noises. So we add a *start* bit in front of the character. Now we must turn off the printer so it doesn't generate "garbage" sensing line noises as character bits. For this we follow the parity, or eighth bit, with two stop bits. Now we have a total of 11-bits making up the 8-level ASCII TTY character, as shown in our earlier diagram.

Note that each bit is 9.09 msec wide, making a total character length of 100 msec. The start bit turns on the printer, the first seven bits represent the character—either letter, numeral, control function, etc.—and the stop bits insure the printer stops and resets for the next character.

The parity, or eighth bit, is used to "test" the transmission when there is the possibility noise can obliterate, or add, a bit. The keyboard can be programmed so the parity bit is added to provide odd or even parity; that is, the *total number of character bits* from one to eight is even or odd.

Assume the keyboard is programmed for even parity and the letter Q is transmitted. The letter Q uses three bits—Numbers 1, 5 and 7. Three is an odd number so the eighth bit is automatically added, making the





total number of bits transmittted four, an even number. At the receiving end only the first seven bits are processed, giving the character Q. A *parity* detector, however, counts all the bits; if it's not an even result the printer might indicate an incorrect character has been received, or the parity detector might light a lamp or sound a bell to indicate something is wrong with the transmission. If the character originally had an even number of bits, such as the letter V which uses bits Numbers 2, 3, 5 and 7, the eighth parity bit is not added since the transmission has an even number of bits for the character, and the parity detector "sees" an even count. Both the keyboard and parity detector can be programmed for even or odd parity. Where there is a direct connection to the computer, or a circuit with little likelihood of noise, the parity detector is usually disabled or simply not used.

Though a TTY requires a bit length of 9.09 msec. the resultant 110 words per minute printer speed is too slow for most users. 300 WPM is the standard for electronic terminals used by time-share (with echo) and personal computers. We get the higher speed by simply reducing the time of each bit to 3.3 msec., and by eliminating the second stop bit.

The actual TTY transmission of the No. 8 is shown in the topmost figure. Note that the reference is *mark,* and the transmitted bits are *start,* 4-5-6, stop-stop. The middle figure shows the electrical equivalent as would be transmitted by an electronic keyboard using the EIA's RS-232 standard. Note that *mark* is now "O" (zero) and the waveform appears as a conventional waveform with the bits positive-going. (Yes, RS-232 provides for a negative-going waveform resembling the TTY waveform shown in the topmost figure). Note that the RS-232 character shown is 300 *baud* so the second stop bit is eliminated.

In both the top and middle figures bit number 1 is not used so the start bit stands alone. The bottom figure shows what occurs when bit 1 is part of the character. The letter "U" has alternating bits starting at bit 1, so bit 1 combines with the start bit to form a "pulse" double the reference bit length of 3.3 msec. The receiving device senses the leading edge of the start bit, times out for 3.3 msec. and since it "sees" a signal at the start of the next 3.3 msec. interval it "counts" bit 1. It is through the timing that the receiver counts bits. For example, in the top figure bits 4, 5 and 6 run together and "appear" as part of a square waveform. In actual fact the TTY doesn't recognize a square wave; its timing mechanism simply counts three contiguous bits followed by a space.

In both TTY, modem, and most personal computer use each ASCII character is transmitted *serially,* meaning one bit follows the other. There are special circuits, however, that require a *parallel* transmission, that is, all bits are transmitted through at least seven wires at the same time (eight wires if parity is used). This is generally done electronically with each bit represented by a *high* (logic 1).

The subject of the ASCII code and its transmission is rather extensive. We have featured the general highlights you'll need to know as you get started in using computers. A more complete explanation geared for hobbyist use can be found in the Sam's publication **TV Typewriter Cookbook** by Don Lancaster. If possibe, avoid the TTY service manuals and handbooks; you need a whole set to find out what's going on and the Cookbook is faster and more easily understood.

terminal are at a different frequency than those from the computer.

The basic input/output device for both time-share and most personal computer systems is the model 33 teletype, and the terminology for the model 33 has become, more or less, the accepted terminology for all I/Os. A model 33 printer (no keyboard) is called an RO for *R*ead *O*nly. CRT displays, or any other type of printer without an associated keyboard are usually termed ROs.

A model 33 teletype which consists of a printer and keyboard is termed a KSR for *K*eyboard *S*end and *R*eceive. Any other type of TTY or CRT terminals with a keyboard and display is termed a KSR.

Then there is ASR for *A*utomatic *S*end and *R*eceive. This is a model 33 TTY with an attached paper tape punch and a reader to read back the paper tape. The tape can be punched either from the keyboard in the *local* operating mode, or by the computer via the echo. The punch is mechanically attached to the printer so whatever the printer receives (with some exceptions programmed through a "stunt box") gets punched out on the tape. The reader feeds out as does the keyboard. If the tape is played (or *read*) by the reader, the output from the TTY is just as if it was sent from the keyboard. Both the punch and reader can be controlled from the computer by special signals, hence, this type of TTY is termed Automatic Send and Receive, or ASR Any similar terminal device is known as an ASR.

Though the model 33 TTY, and its designations such as RO and KRS, is the common standard of reference for most hobby and time-share computer systems there are other keyboard-printer devices that provide similar functions. Other straight TTY equipments are available as ASR and KSR equivalents of the model 33; and there are special *line printers* that print an entire line of copy at high speed, in contrast to the TTY's one-letter-at-a-time printout.

Smart and Dumb. Another variation is the "dumb" CRT terminal. This is a keyboard and CRT display made to function exactly as a KSR TTY. Though all-electronic, it puts out exactly the same type of signal as a TTY and responds to the same input signal, printing one letter at a time at TTY speed on the CRT. A slightly different version of the KSR electronic TTY puts out something called an RS-232 signal, an industry-wide TTL (transistor-transistor logic) compatible signal used for direct connection to computers and some modems; but it's still a "dumb" terminal.

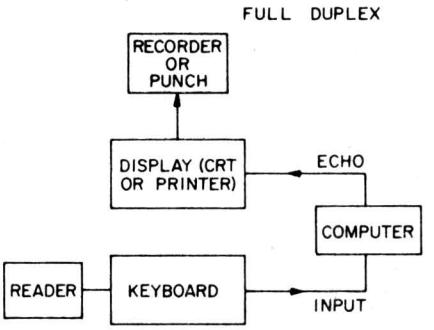
PHONE THE FUTURE

What's a "smart" terminal? An electronic terminal with a built in minicomputer. A smart terminal allows the user to type a full page into storage, examine the page for errors, make corrections, and then transmit the signal. In some smart terminals additional information can be typed into storage as the previously entered information is sent out. On the receiving end, a smart terminal can often store several pages of transmission, allowing the user to electronically roll back to previous pages. (A full screen is usually a page.) Some smart terminals are actually full computers with built in KSR and can store thousands of words, but this is esoteric equipment. The typical hobbyist and time-share user is more likely to use an inexpensive, dumb KSR electronic terminal such as the highly rated Micro-Term ACT-1, which has 300 *baud* (we'll explain *baud* later) RS-232 output that can be fed directly to a hobby computer or into a modem such as the Omnitech 701A (the industry "work horse" available used for about $150) that has both a TTY and RS-232 connection.

For those who need both a direct RS-232 output and a modem, but who don't want the modem as an extra piece of gear with its required wiring there is the ACT-2 terminal shown in the photographs. The one shown is a prototype sent to our laboratory for extensive testing and evaluation. It is essentially the ACT-1 with a built-in modem.

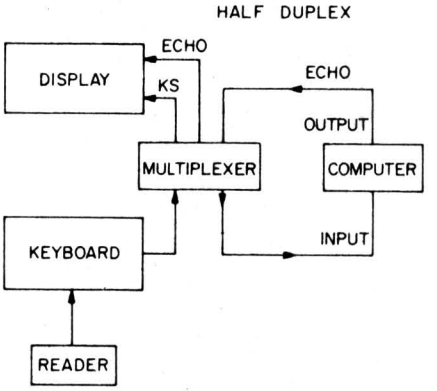
Baud. Whether you go into personal or timeshare computer systems you're going to hear and read much about something termed *baud* or *baud rate*. Very simply, *baud* is expertese for something we used to term *pulse width*, or *time;* it's sort of like when the experts decided to upgrade electronics by deciding we must all call cycle-per-second *hertz*.

Here it is—the standard of reference for many time-share and personal computer installations. This model 33 teletype is used whenever hard copy, a written record, is required. The model pictured is an ASR—Automatic Send and Receive. Visible to the left is the paper tape punch and reader.



For computer use a terminal—whether TTY or electronic with CRT—is generally arranged in the full-duplex mode, meaning the keyboard and display are independent of each other. In some instances the keyboard will have an associated reader that can provide output from a punched tape (or other player), while the display has an associated recorder, or a punch for paper tape. If there is an associated reader and punch the complete terminal is an ASR (Automatic Send and Receive).

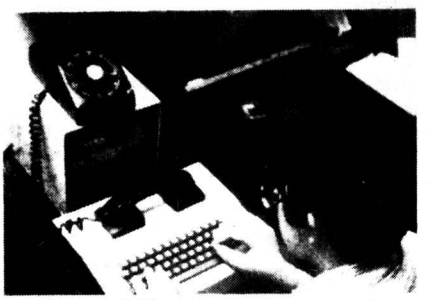
The keyboard sends the characters to the computer which in turn sends back an instantaneous echo to the printer/display. It appears to the user that he is typing directly to the display though there is, in fact, no direct connection between the keyboard and display.



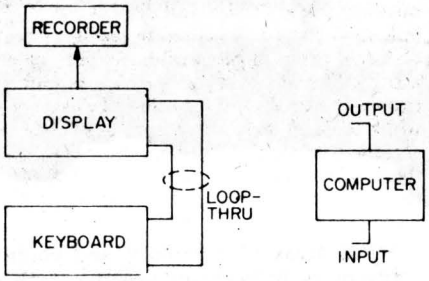
In the half-duplex mode a form of multiplex device somewhere in the overall system—it can be a simple wire from the keyboard—sends the keyboard signal to the display even if the computer's echo is turned off. If the echo is left on the display will show each character twice, once from the direct connection from the keyboard followed by the echo. Half-duplex is generally used when the terminal operates at baud rates too fast for the printer to follow the echo, or too fast for the computer to handle the incoming character plus the echo. In time-share and hobby personal computer systems the echo is usually turned off at baud rates higher than 300.

In order for electronic systems to communicate there must be some form of synchronization; each character or command must be transmitted and received in a specific time interval. If a printer was programmed to receive a character in say, 100 milliseconds (msec.), and it received one and a half characters in 100 msec., obviously it would be confused and display incorrect information, just as it would if it received only half the character information in 100 msec.

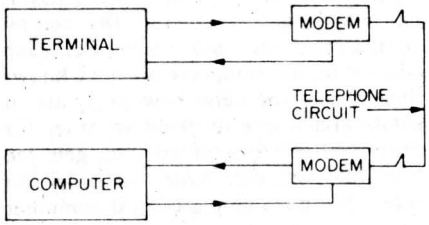
As you might have guessed, we achieve synchronization by transmitting any character in a specific time interval. If you refer to the illustration of an ASCII TTY signal you'll note it consists of 11 bits—the start bit, 7 character bits, parity, and two stop bits—with each bit being precisely 9.09 msec. in width for a total of 100 msec. Now *baud*, or *baud rate* is very simply the reciprocal of the pulse width of each bit. Since the pulse width of each ASCII TTY bit is 9.09 msec the *baud rate* is 1/.00909, or 110. That's all there is to it. We could just as easily



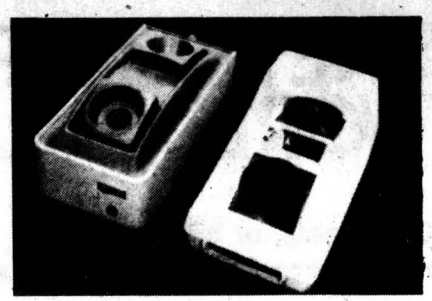
Nothing could be simpler than using the ACT-2's coupler. Just place the telephone handset in its niche and have full access to such systems as Call Data which accommodate virtually all programming languages. Call Data, for instance, can do such things as feed back a cross assembler for your personal computer, or provide access to the Dartmouth University program library.



When a terminal is switched to the local mode the keyboard is connected directly to the display via a loop, and the computer is disconnected. In this way the keyboard entries are fed directly to the printer and tape punch of an ASR TTY, and tapes can be prepared for feed to the computer at a later time. It also permits characters such as control functions to be added to a punched tape that has been previously made from a feed by the computer using the full or half duplex modes. (In half-duplex the computer can feed the display if the keyboard is not in use. This is usually accomplished by sending special control signals to the computer.) In electronic terminal systems with CRT display the recorder might well be a digital tape recorder rather than a TTY-type tape punch.



A terminal can be connected to a computer through voice-grade telephone circuits using a device termed a modem, an acronym for modulator-demodulator. The modem converts the terminal's electrical pulses to tones in the approximate 1000 to 2000 Hz range. When the tones are received by the computer's modem they are converted back to electrical signals. Each modem is a two-way device: the computer's modem converts the computer's signal to audio tones and feeds it to the terminal's modem where it is converted back to an electrical signal for the printer or display. To avoid confusion between the terminal and computer the tones from the terminal are completely different from the tones generated by the computer's modem.



Here we have both the accepted standard of the industry and a promising newcomer. The modem on the left is the Omnitec 701A and originally cost more than $400. It handles both TTY and RS-232 inputs and can be directly connected (hard wired) to the phone line; it also provides half and full duplex switching. Used ones can be had for about $150. On its right is the new Omnitec 501A—specifically made for the model 33 TTY. It is available fully assembled for $150, and as a semi-kit for $100—all PC board wiring complete, it just needs to be placed in its cabinet. It too provides both half and full duplex. Avoid hobby-type modems which do not have a duplex switching option directly available on the front or rear.

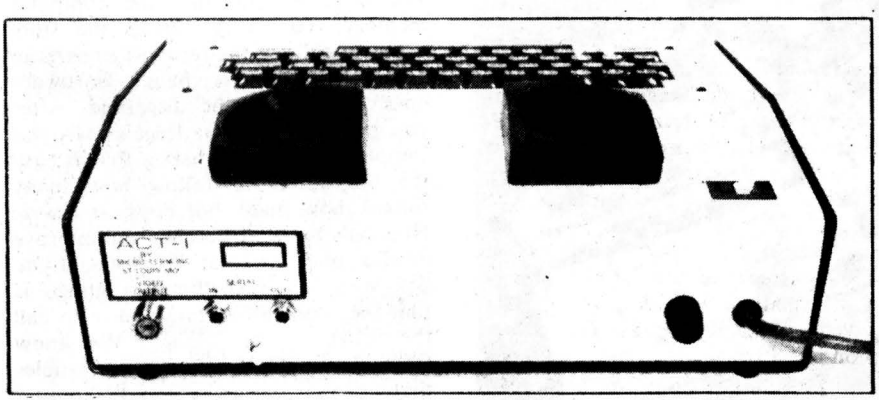
have called it the "Irving rate," the "Gloria rate," or the "9.09 rate." We call it *baud rate*, and a 110 baud rate is always 8-level TTY speed.

Trouble is, the TTY is capable of printing a maximum of 100 words per minute typing rate, which is fine for someone typing away, but slow when getting information back from a computer.

Though there are printers capable of displaying more than 100 words per minute they cannot print faster than the information is transmitted, and the limitation remains the speed at which the ASCII characters are transmitted. But, the ASCII format applies to the sequence of the eight bits making up the character and parity (a self-checking feature). If each bit were shortened each character would take less time to transmit, and that's exactly what is done. By general convention, for most personal and time-share *high speed* terminals the character bit rate is shortened from 9.09 msec to 3.3 msec., Almost three times as much information may be transmitted within a given time interval as compared to a 110 baud TTY. If we find the reciprocal of 3.3 msec. (1/.0033) we get an answer of 300, meaning a 300 *baud* rate. You will find most hobby CRT terminals such as the ACT-1 and ACT-2 to be 300 baud.

The fact that 110 baud means 100 words per minute and 300 baud means 300 words per minute doesn't mean there is a direct relationship between baud rate and words per minute. The close relationship is purely accidental. (If you get out the calculator and start multiplying you'll probably claim a 300 baud rate should produce less than 300 words per minute. The "error" comes in because the TTY signal has two stop bits which take up a total of 18.18 msec. The complete TTY signal is 11 bits; start, 8 characters with parity, 2 stops. The 300 baud system has only one stop bit—ten bits total for the character—so the stop is 3.3 msec,

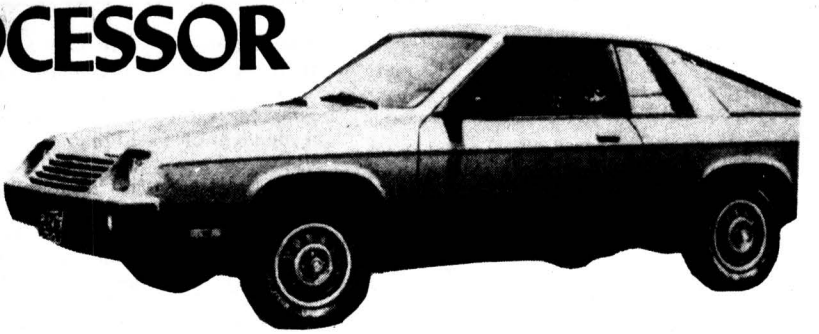
(Continued on page 81)



This actual prototype of an ACT-2 uses an older ACT-1 (hence the label) with a built-in modem for interfacing to the telephone lines. The video output feeds a TV monitor (CRT) and the two pieces make up a complete terminal. Inputs and outputs are RS-232 and when phone plugs are inserted the modem is automatically disconnected. A switch for full duplex or half duplex is located directly under the phone jacks. Using the modem is just a matter of pulling the plugs and then placing the handset in the coupler.

THE MICROPROCESSOR WORLD

by Roy Adams



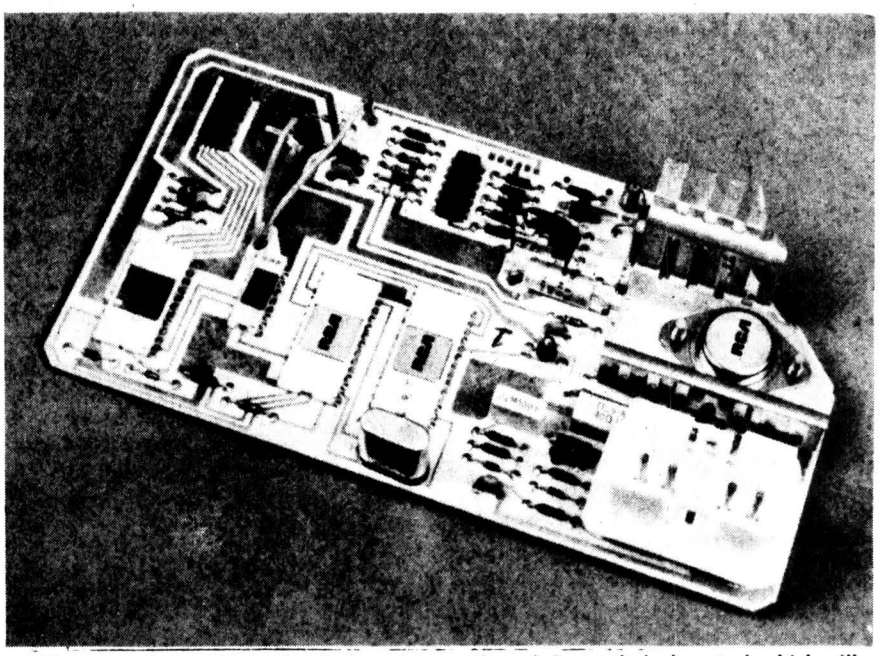
On wheels, or in the kitchen, microprocessors are here to stay

☐ Our world has already changed because of microcomputers, and it will change much more. In ten to fifteen years, your home will likely have many microprocessors in it. They will be in your kitchen, garage, basement, living room, gameroom, and bedroom. Many people will be using microcomputers without knowing it. Others will be expanding their thinking processes as they pit themselves against complex but fascinating learning machines. Math, science, even history and art will be programmed into shoebox sized computers that you can buy or that you can borrow from a library or school.

We are talking about a subject as broad as your imagination—the applications of microcomputers. Let's face it, your goal and that of every electronics outfit in this business, is to first understand some principles of microcomputers and then to figure out what to do with them. You may be a hobbyist who is simply curious and trying to expand his horizons. But, you may be more than that. You may want to start your own business to make and market a product of your own invention. The time has come to understand what is being done with microcomputers, what some companies are thinking about doing, and to pipedream a bit about how we can improve our world with some futuristic inventions.

The area of microprocessor applications is extremely broad, and in this article we can only begin to cover it. In fact, we'll confine ourselves to microcomputers in business, transportation systems, and the home. But the home applications will only be touched on, because it's so broad. After all, home uses include games, and we all know how many games for microcompressors and TV sets there are around.

Computers in the Office. The use of computers in business can be broken up conveniently enough into two broad areas of inventory and control. Inventory here means keeping track of something. It can be the recordkeeping of a doctor's office or of an accountant for billing purposes, or it can be keeping track of what items are in stock. Let's take a look at some actual cases. There is, of course, the one most of you have seen by now in some of the grocery stores. A light pen reads the international strip code from a label, enters the code into the cashier's terminal which passes it onto the central computer in the back room. For each label code, the computer memory knows the price of the item, how many are in stock, and where to place an order for more. When stock of an item gets too low, the computer writes an order for more. We have here a central computer controlling small microcomputers in the cashiers' terminals. Commands flow back and forth to control the light pen, give pricing, tax computation, *et cetera.* Then there is your local hamburger heaven. Have you noticed the key pad on a Burger King cashier's terminal? Only a microcomputer can provide the kind of flexibility you see there. You say hamburger, for example, and the order taker presses a touch-type key with a picture of a hamburger on it. The price is built into the computer memory. To change prices the store manager calls in the terminal experts to do a simple software change. Hardware does not have to be discarded. After you have placed your large order, this computer provides a listing that is easy for the cooks to follow—how many shakes, how many hot dogs, *et cetera.* Now let's think of how we can improve on this smart terminal. What would you do? Well, the first thing to notice is that the terminal operators have to call the order to the kitchen. We know from all the TV and magazine articles that video screens are a blooming output medium. It seems archaic to go from a computer to a person to a microphone to the kitchen. The kitchen should perhaps have a video display that shows the hamburger person how many burgers he has on order of various types (small, large, no pickle, etc.). As the



This PC board assembly is Chrysler's next generation of electronic fuel control, which will serve as an alternative to its present lean-burn system. The four dual-in-line packages on the lower left of the PC board contain the heart of the system, (left to right) the CDP 1833 ROM, the CDP 1824 RAM, a custom-made CPU and a custom input/output port.

outgoing tray is filled, the computer would subtract the items from computer memory card from the screen. Inventory ability could be added by connecting the terminals to a central inventory control computer that ordered food as necessary via a data link to the warehouse computer. Eventually, you could build an entirely automated entry. Drive in, push the key for the food you want, the kitchen would be composed of conveyor belts and ovens—out comes your food.

That is a bit astray from pure inventory control, but that kind of leapfrogging is exactly what is making small computer businesses take off to big things today. Inventory control is very much in demand, and every case has to be nearly custom tailored to the client. Through software control, that customizing is no problem. If you want to imagine how you could get started with an inventory control system, simply imagine getting a small computer that takes BASIC instructions (BASIC is the most popular of the home computer languages) via a typewriter keyboard, and next imagine programming a small and simple inventory system to keep track of shoes in a store. You are on your way.

. . . And In The Factory. Computer control in industry is presently widespread, but the potential is really unlimited. The needs are great, and the microcomputers available today literally sit waiting for someone to apply their power to the jobs. Take, for example, the computer-automated control of metal parts. Machines that stamp or bend metal parts are becoming nearly commonplace in industry. The operator loads a hopper with pipes or other metal pieces, presses the button and the computer controls the machine. Out come car ash trays, hingles, you-name-it. If a new hinge design is to be fabricated, the computer program is simply updated with new holes to drill and bends to make.

The ideas here are not new, and in many cases only the surface is being scratched. Microcomputers of the futur will be tied together in an industrial plant. The ordering of raw material to meet schedules on new orders, the actual fabrications, and the shipping and billing would all be computer controlled. And remember, you need not sit by and watch. There is a big market out there and learning about microcomputer operations as a hobby at home will give you a big foot in the door.

The Millers Ferry Hydroelectric Plant in Alabama is controlled by an Interdata 70 that communicates with three IMP microprocessor boards to control water flow and generators at Jones Bluff Dam that is many miles away. The system is estimated to save tens of thousands of dollars by increasing efficiency of the power plant—so the computers will pay for themselves. Control of open hearth furnace feeding and temperatures, monitor and control of electric power to giant aluminum smelting pots, monitoring of the quantity of ingredients that go into anything from cake mixes to tire rubber—all of these use microcomputers today. The computer acts as a kind of central brain that may have several smaller microprocessors feeding it, which in turn have sensors telling them what is happening. So sensor technology is an area that is absolutely booming as computers push the need for new and cheaper devices. Sensors can smell (such as smoke detectors), feel (like strain gauges), hear (special frequency response transducers), and see (like infra red sensors). Still, there is the need for more. The more intelligence that can be built into the sensors, the less time the central computer has to spend interpreting.

In Your Car, Too! Transportation is an area much in need of micro-processors—and the future will show a fantastic set of changes in this area. The Chrysler Lean-Burn system is one of the popular examples of how people can have a computer in their garage without knowing it. The concept there is to control the spark firing so as to always have the air/fuel ratio equal to 18 to 1. While maximum fuel economy is achieved at a ratio of about 16.5 to 1, a ratio of 18 to 1 leads to minimum levels of carbon monoxide and unburned hydrocarbons. In 1981, nitrous emissions will be tightened from 2 grams per mile to only 1 gram per mile. Several manufacturers plan to use a new type catalytic converter to absorb the bad exhaust, but for that new converter to work well, and for fuel economy to be kept as high as possible, a more elaborate computer control system is being developed. ■

MICRO PROCESSOR USES IN AUTOS

Fuel Economy and Emission Control

- Firing of plugs
- Air/fuel ratio
- Rate of deceleration
- Speed control for highest m.p.g.
- Automatic adjustment for weather conditions

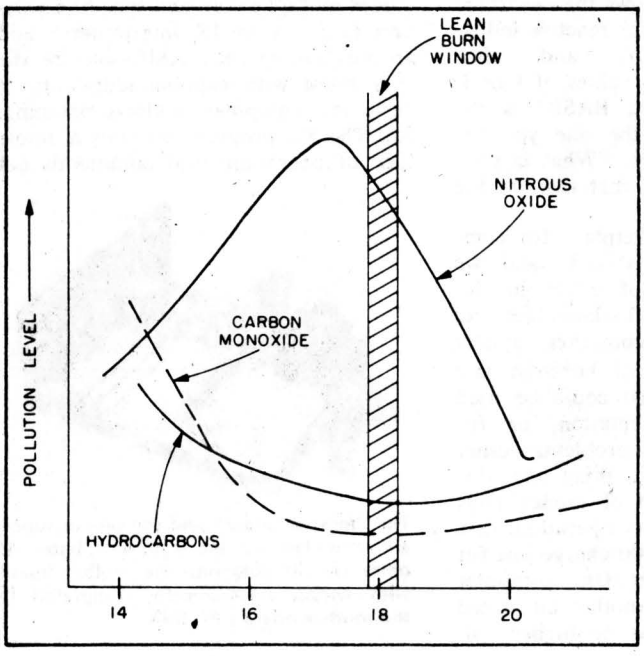
Driving Aids

- All electronic instrument panel
- Radar with speed control
- Computer navigation following road beacons
- Computation of best rest stops, expected arrival time, type fuel to use for trip, best tire pressure for load, etc.

Safety

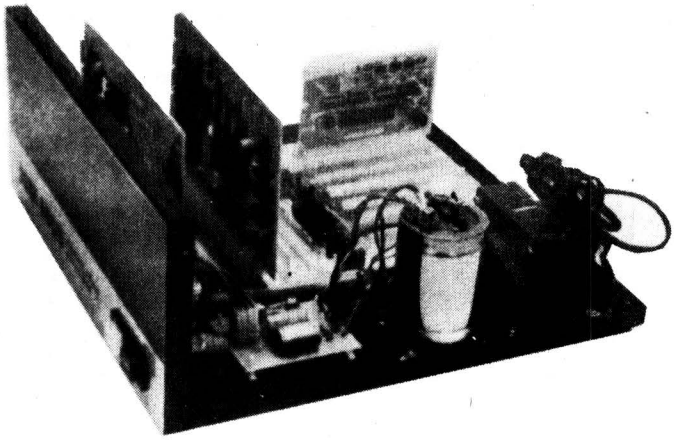
- Computer warning of weak spots like tires, battery, brakes, radar, etc.
- Anti-skid braking
- Anti-theft computer combination locks
- Air-bag control

Microprocessors can be especially useful in aiding operators of complex equipment, such as the family car. The auto industry hopes to perfect all the above uses of microprocessors in the coming years. "Dumb" cars will be in museums instead of on freeways!



A lean-burn computer keeps the air/fuel mixture at 18:1 under all driving conditions in order to keep pollution levels low. A new computer system must be devised by 1981—when Congress will cut allowable nitrous oxide levels by fifty percent.

HCH checks out the...

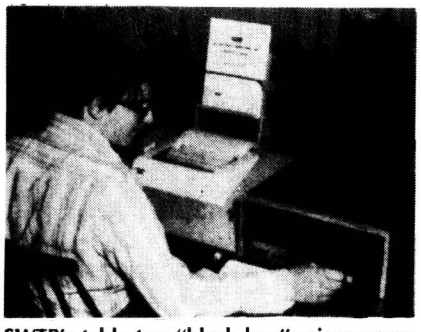


SWTP 6800 Computer

When the computer bug *bytes*, this new unit is the right Rx.

☐ It seems so easy. Just send for a box of parts. Spend a few evenings putting everything together, and when you're all done you have a real computer; a computer powerful enough to equal many time-sharing systems owned by some small school districts and colleges. The price? Less than that of a decent—not great—high fidelity system. Anywhere in the range of less than $400 to under $1000.

Only problem is, a computer by itself can do absolutely nothing. It just sits there on the table. Possibly, it might have a row of switches, and lights that wink and blink depending on the operation sequence of the switches, but for the average electronic hobbyist it's useless. There must be some way to talk to the computer, and for that we need a terminal, an input/output device—or I/O as it's more commonly termed. Cheapest I/O is a CRT video display terminal for about $400 in kit form, $550 wired. Now we're up to at least $1000, but all we have is a system that could be used only by someone with considerable training in *machine language*, generally college level work.



SWTP's table-top "black box" microprocessor computer and a terminal are all that's needed for a personal computing system. Simply press the white reset button on the front panel and you can program directly from the terminal—in this instance a model ASR 33 TTY.

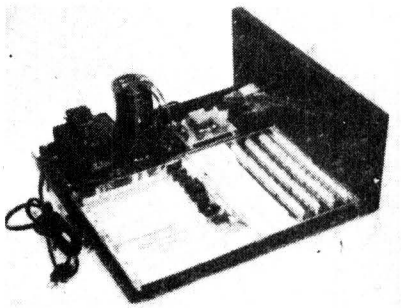
Of course, most computer advertisements make it look easy: "Own your personal computer. Play *Star-Trek*, Tic-Tac-Toe, etc." If you believe a few games are worth $1000 or more then latch onto any personal computer kit.

But, if you want to be able to use a computer for serious work without a degree in computer science, then you need one easily and inexpensively programmed in a "higher language." BASIC is such a program. A BASIC-equipped computer and terminal reads out READY when it wants you to enter information. BASIC allows even children to build simple programs such as A + B = C. It allows the electronic hobbyist to program for reactive inductance—X_L=6.28(f)(L), and then runs a hundred or so values of f or L in a minute. In fact, BASIC is the beginner's language, the one you see people using that asks: "What is your name?" The language that *talks* to the user.

Surveying the marketplace for computer kits is no easy task, and we took a great deal of effort in doing so. We were looking for an easy to assemble computer system suitable for the typical hobbyist and student, something that could be used for general experimentation, or for school or business problems other than electronic. First thing we discovered, is that a lot of money buys little in the way of easy operation. One computer kit has a $200 charge just for the BASIC interpreter (the computer language program). Another kit priced in excess of $500 has no higher language available. Yet another $500 kit has a "Tiny BASIC" good for a few small problems and games (games again).

A Fine Kit. When we were done we came up with a computer system *kit* which we presently consider a "final buy" for the average hobbyist, the Southwest Technical Products Corp.'s Model 6800 Computer System.

The reason we've selected the SWTP 6800 is twofold. Firstly, it has a built in *resident monitor* that eliminates the need for panel swtiches and lights. You don't need to know machine language to use the 6800; because of the monitor you can enter data in hexadecimal form directly from the terminal. But even more important, SWTP has a really excellent 4K BASIC interpreter that includes multiplication and scientific notation (a rarity in 4K interpreters) and an outstanding 8K BASIC—one of the very finest with exponentiation, direct peripheral equipment address, patching, etc. The 8K program contains a whole slew of operations and commands not



The "mother board" and the power supply are mounted on the cabinet's base. All other circuits plug into the mother board, with wiring automatically completed by the mother board's PC foils.

SWTP 6800

found on some slightly older time-share BASIC systems. The 4K and 8K BASIC interpreters for the SWTP 6800 computer are available on TTY tape and cassette from about $5 to $20 depending on the program and its base (paper tape or cassette). Though there are other 6800 type computer kits available, it's SWTP's BASIC programs that makes the SWTP computer our "best buy."

Before going further, let's explain the whole bit about 4K and 8K memory systems: The "K" stands for a thousand *bytes*—eight bit "words." A computer with 4K memory RAM (random access memory) can be programmed up to, or has the capacity for, 4000 bytes. An 8K memory means 8000 bytes Theoretically, 4K BASIC should fit into a

This is what a 4K memory board looks like with only the ICs for 2K of memory. We've left the other 2K of memory chips out so you can see the extensive use of sockets. Contrary to what SWTP says, if you don't want heartaches, use sockets for all ICs. Only the voltage regulator, which must be secured to a heat sink, does not have a socket.

4K memory. Unfortunately, it doesn't. 4K BASIC actually needs about 4.5K of memory; so a computer must have about 6K of memory in order to program using 4K BASIC. On the other hand, 8K BASIC needs only 6.5K for the interpreter; so 8K of memory handles 8K BASIC very well. Since 4K of memory cannot handle 4K BASIC, most users of personal computers with 4K memory use a smaller version of BASIC known as "Tiny BASIC," which is good for little more than playing games.

For the average hobbyist, we don't believe a computer system should be limited primarily to games. We strongly suggest the SWTP 6800 computer system should be obtained with at least 6K of memory.

The stripped down SWTP 6800 computer kit consists of a "black box" ($439) with just an illuminated power switch, a reset switch, a mother board. a microprocessor/system board, a 4K memory board, and a serial control interface board that can be programmed for either a 110 *baud* TTY or a 300 *baud* CRT terminal. The power supply components mount directly on the base of the "black box" cabinet. As assembled by the user, the mother board can accommodate five additional 4K memory boards (under $100 each) for a total of 24K of memory, and seven more interface boards for a total of eight peripheral I/O devices such as line printers, recorders, etc. You simply plug in the optional memory or interface boards

Construction. The PC boards are first quality, and the ICs had a notably excellent failure rate—only one was defective. This, considering the large number of ICs supplied, is excellent. (We have come to expect up to a 10 percent IC failure rate in kits.) To successfully build this kit you *must* use a needle pointed soldering iron rated about 25 watts. Anything larger will result in solder bridges—the printed foils are unusually close compared to the usual hobbyist kit's printed circuit board(s). Also, you must have someone well skilled in computer technology willing to help you debug the system; we'll explain the need for this later.

Unlike a Heathkit, SWTP's instructions don't spell out every step. You get a rather good pictorial (two color) for each board, a separate parts list, a set of general instructions, and you use all three to determine what part goes where, and when. The component packaging is unusually good. As a general rule the components are packaged for a small number of steps; when you complete one package you (generally) move on to another package. (All the components are not jumbled together—a welcome variation on the kit packaging we've been getting from the "biggies" the past few years.)

Warning. SWTP does not supply sockets for the ICs; they specify the ICs should be soldered to the PC boards. Whatever you do, don't make this foolish mistake. USE SOCKETS FOR EVERY IC. If you wind up with a defective IC (as we did), or intáll an IC backwards (which is very easy to do) you'll make hash out of the board trying to remove the defective or reversed IC. A complete set of sockets for the basic 6800 computer kit is available from the Computer Mart of New York, 118 Madison Ave., N.Y., N.Y. 10016, for $21.70 post paid. They also have socket sets for the optional SWTP computer boards (such as additional 4K memory boards).

Do not handle any IC unless you're grounded just as specified by SWTP. They even provide a 1-megohm resistor so you can "safety-ground" from your wrist to the electrical ground. A wire to the ground bus or chassis of the computer is not a ground. *You must be grounded to the electrical ground.* Any tool or knife used near the ICs must be grounded. While not all the ICs are prone to damage by static electricity, the C-MOS types are, and it takes just a flick of the finger to zap an IC. So don't open any IC packages until told to do so, and don't open or handle the ICs until you're properly grounded. If your soldering iron doesn't have a three-wire (grounded) power cord never apply the iron to the PC board(s) once the ICs are installed.

Testing. If you're extra careful, and

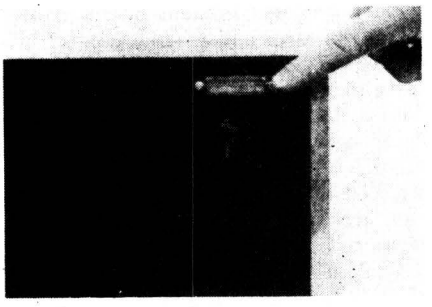
These three boards, which plug into the mother board, make up the basic computer. The central processing board is on the left, the 4K memory board on the right, and the serial input/control interface is in the center. Additional memory and interface boards can be plugged into the mother board.

double-check every step, the computer kit should take a couple of long evenings, maybe ten hours. To check out the computer you will need some form of terminal. Many of you will not understand one word of the rather extensive check out and programming instructions, that's where someone who has taken a computer science course comes in. He can run the diagnostic programs specified in the instruction manual. These diagnostics can pinpoint almost any defect, whether a defective component or a wiring error If you have purchased the kit from a computer specialty store, such as one of the Computer Marts, they might check out your boards in their own system; it depends on the particular store and the guy running it. If you make a botch of the assembly, some computer specialty shops will do the service for you at a modest fee. A friend who has studied computer science, or the specialty shop, are your two best bets if you have problems. SWTP tries hard but they're just not geared for extensive service. They will supply parts by mail rather quickly, and try to answer your letters. But, getting them on the phone can be rather expensive at long distance rates. That's why we suggest you latch on to a college student before you tackle a computer kit, or deal with one of SWTP's major dealers (the price is the same from SWTP or a dealer).

The only problem we had with the basic computer kit was a defective IC—thank heaven we used sockets!

When completed, the computer kit had 4K of memory, which could do very little other than run the diagnostics and a few simple programs and games (those games again). So, we purchased an optional 4K memory board, giving us a total of 8K memory, and we loaded in a 4K BASIC tape, and found we had no multiplication—the tape was defective. After getting a good 4K tape we had a flexible language suitable for high school level math, and many electronic programs. After a long wait, because the demand exceeded the supply at the time, we got our 8K BASIC tape which opened up a whole new world. The SWTP 8K BASIC is a powerful interpreter allowing such things as trancendental functions character strings, and user defined functions. Fact is, the SWTP BASIC is much more powerful than the BASIC in the time-share system of our local central high school district, and they have a notably good computer system and course.

The SWTP 6800 kit is a very good choice for a personal computer kit because of its relatively low cost in



Connections, from the terminals and other peripheral equipment, are normally made through grommeted holes in the rear of the cabinet. We have installed a Type-D connector so with proper jumpers in the associated plugs the single serial/control interface can be used for 110 or 300 baud, and TTY or RS-232 input/output.

relation to other computer kits, its built in monitor system (Motorola's MIKBUG) and a notably excellent BASIC interpreter. The system's only problem is common to all presently available personal computers: you lose the programs when the power is turned off. Either SWTP or an accessory supplier will eventually come out with a non-volatile extended BASIC (8K or greater) in PROM, ROM, or EPROM; and, there is word that someone is working on a FORTRAN compiler for the 6800 system.

Getting Started. As you have surmised by now, you need a lot more equipment than "just a computer" in order to have a computer system. For starters, you must have an Automatic Send and Receive Teletype (ASR TTY) so you can feed in program tapes: if you can't feed in long programs by tape you're back to games. Alternately, you might prefer a CRT TV terminal, in which case you will need either an optional, punched tape reader, (a reasonably priced accessory), or a cassette recorder and an interface between the recorder and computer (cost slightly greater than a paper tape reader, and you must build the interface. Programs are available on cassette tape for slightly less than the cost of the same program on punched paper tape).

You need the 6800 base computer, an additional 4K of memory, a terminal, some form of paper or cassette tape reader, and a BASIC interpreter program if you plan on anything more than playing games or writing simple programs.

Often, if you make arrangements to get all your equipment from one source in a package deal, some items will be thrown in at a "discount" or "free." After all, computer stores are still just stores, and their primary concern is still to keep the customers coming in and to keep them interested in buying from them. So expect a lot from them.

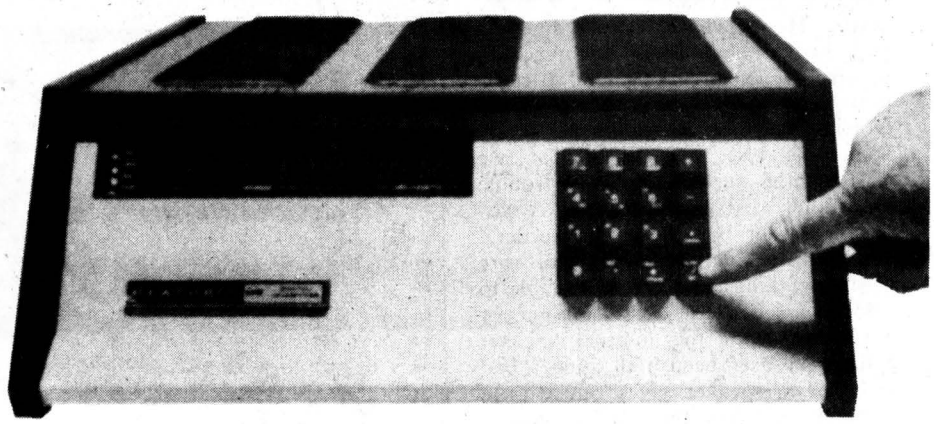
We suggest you purchase your computer hardware from a local dealer because he will give you some degree of free or low cost assistance if you have problems. While our problem was only a single defective IC in the kit, it might have been a real hassle to handle it by mail or phone. Further, when we found we had a defective 4K tape our local dealer swapped the tape on the spot.

Would We Do It Again? The answer is yes. Looking back at our problem(s), the actual construction of the kit, and the final result in respect to the 8K BASIC available from SWTP, we would start out in personal computing exactly the same way, with a SWTP 6800 computer kit. The only change we would make in our system would be to save space by using a CRT TV terminal instead of a model 33 TTY, and we would use an inexpensive optical reader to feed the 8K BASIC tape into the computer. We would sacrifice the "hard copy" of the TTY for the compact size of a CRT terminal, adding a line printer for hard copy at a later time. But, all in all, we find we are using the SWTP 6800 computer in preference to our time-share system; in the long run it will work out less expensive than paying hourly charges for time-share computing. ■

Our college computer technology student was Morris Balamut, who is also the manager of the Computer Mart of New York. This is our memory board he's plugging in to the store's own SWTP 6800 to check for proper operation one board at a time). Turned out we had a defective IC. We strongly urge you do not tackle a computer kit unless you have a college level computer technology student and/or a local authorized kit dealer to back you up if you have problems. In many instances you'll find that store personnel and college level assistance are one and the same. If you must get service direct from a computer manufacturer you might be in for a long wait. When the price direct or from an authorized dealer is the same (don't forget to allow for shipping charges) get your computer kit from a local dealer.

HCH checks out the...

HEATHKIT H8 HOME COMPUTER



Here's a personal computer you can build yourself for years of computing fun!

RAW, BRUTE COMPUTING POWER is about the best way to describe Heathkit's H8 computer. The H8 is unlike many other personal and hobby computers which leave you hanging in need of an advanced BASIC (or other software); or which turn out to be short on adaptable peripherals such as recorders and printers; or which provide little technical backup in the event you have assembly problems. The Heathkit H8 can meet almost any reasonable software need; can be almost instantly expanded to accommodate any peripheral; and has some very substantial technical backup. Besides, while this doesn't seem important at first thought, the H8 is built like a battleship—it's among the most rugged of personal computers in terms of industrial and/or school use.

The basic package ($375) includes the cabinet, a factory wired and tested 8080A CPU *card,* 10-position motherboard (of which two positions are used by the CPU card and a front panel/monitor), the power supply, a monitor speaker (which we'll cover later), and an *intelligent* front panel that is actually part of a *control circuit board*—which we'll call the CCB for brevity.

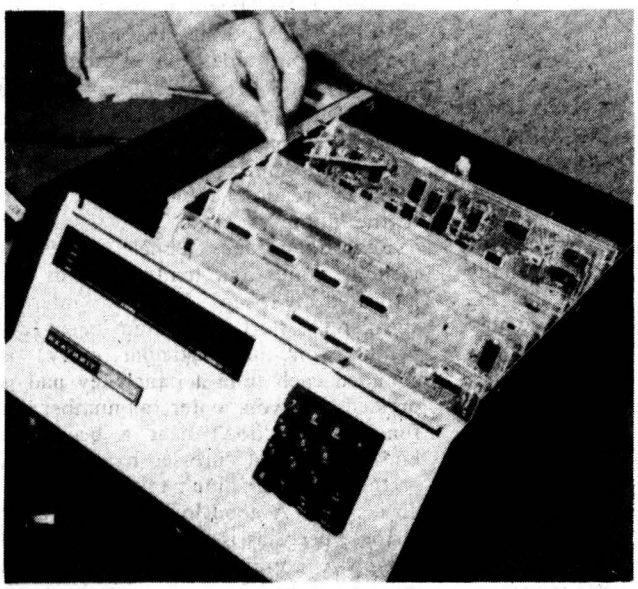
The CCB features a 9-digit LED readout. The readout occupies the front face of the computer and is highly visible. Just by keeping careful watch on the LEDs an H8 user can keep track of important items such as contents of memory addresses; register contents; Input/Output ports; type and condition of dumps and loads; there is even a *checksum* error which helps when you load the computer with software.

In conjunction with the readout is a 16-key pad, also on the computer's front, that provides dumps and loads at the touch of a button. The user can have almost instant selection and data alteration of any memory, register, or the program counter. The counter may be incremented and decremented under manual control; and I/O is under manual control.

Included in the basic package are cassettes with the following software: Benton Harbor Basic; a two-pass assembler (HASL-8); a text editor (TED-8); a console debugger (BUG-8). The resident program monitor (panel monitor) is in ROM (read only memory).

Obviously, you are going to want to use some form of cassette tape machine to feed the software into the computer. The interface for the cassette data system is part of the *optional* H8-5 Serial I/O and Cassette Interface. This cassettte interface operates at 1200 baud (four times faster than the common 300 baud rate used by many personal computers). A switch permits a teletype reader/punch to be used as the data storage device. User selectable baud rates from 110 up permit the use of either a standard teletype terminal or a high speed CRT terminal, such as the Heathkit H9 Video terminal. (We prefer the TTY because it provides a "hard copy.") The H8-5 kit is priced at $110 and plugs directly into the motherboard. (A parallel I/O card with three input/output connections is available for $150.)

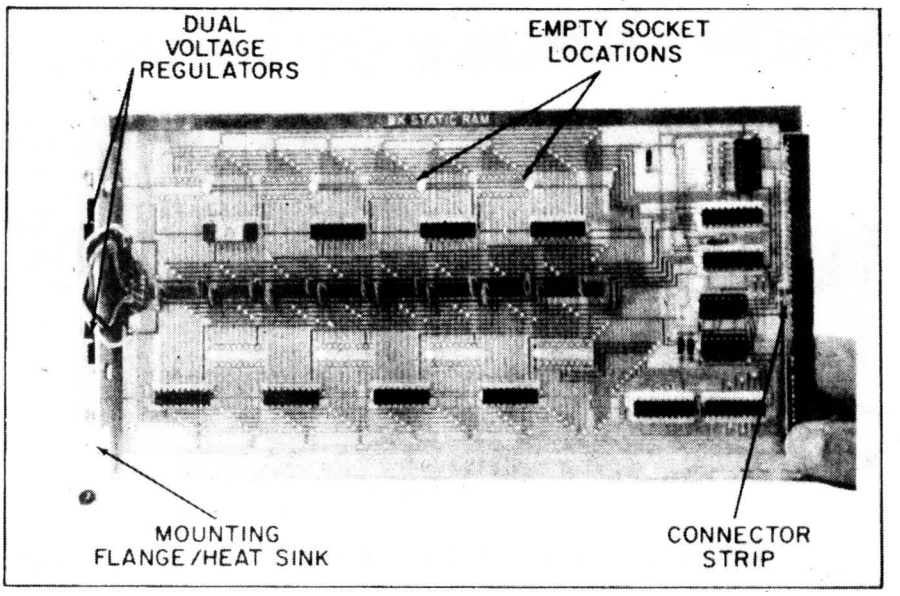
Once you have the H8-5 serial I/O-cassette card you need a cassette re-



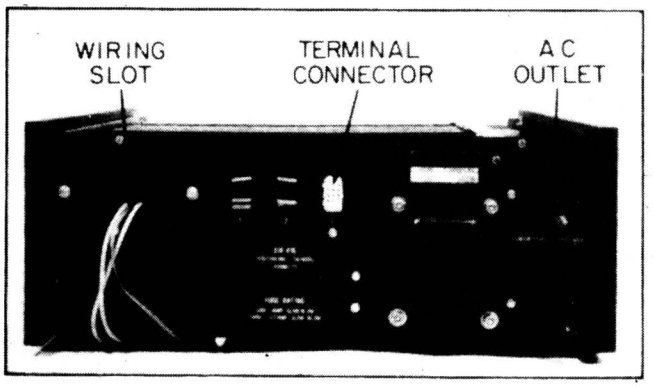
Heathkit's H8 computer will raise you into the elite circles of Personal Computing. Based on the 8080A chip the computer may be configured to fit just about any use in any small computer situation. The H8's intelligent front panel is a real treat to use.

HEATHKIT H8

corder. Here's where you can run into your first problem. Heathkit suggests a relatively inexpensive General Electric portable recorder they sell for $60. Don't waste time and effort by trying something else. We know, for we wasted three weeks looking for trouble that didn't exist. Figuring that "A cassette recorder is a cassette recorder," we tried to load Heathkit's software with everything from a $20 portable to a $500 cassette deck, and all we got for our efforts was the internal speaker going beep-beep-beep, indicating a bad load. (The speaker gives out with a single beep when the load is good.) Obviously the cassette interface was defective so we sent it back to Heathkit. It was returned within five days. Heathkit claimed our H8-5 assembly was perfect, that there wasn't anything wrong with the card. We packed up the card and recorder and took off for the local Heathkit store, whose technician instantly determined our recorder was not recording the 1200 and 2400 Hz interface tones in phase. As simple as that. We purchased a Heathkit ECP-3801 cassette recorder in the store and that



When you buy a 4K memory board from Heath, you're really investing in a full 8K. The 4K boards arrive completely designed for 8K. Just buy the extra chips with their sockets.



Interfacing the H8 to a terminal is a very simple matter once the proper PC board has been built and installed. All connections are right there.

was the end of our dump load problems. Our recommendation: Save time and effort; get Heathkit's tape recorder at the same time you get the computer.

Okay. Now we have an H8 computer, an H8-5 serial I/O with cassette interface, and software. Other than a terminal, all we need is some memory.

Though many programs can run in 3K of memory, at least 8K is required for most software including BASIC. (The Extended BASIC requires 12K, preferably 16K—but more on this BASIC later.)

Memory card kits (4K) are available for $125. An expansion kit available as an option for the basic 4K kit takes it to 8K for an additional $85. (The basic card is for 8K memory though sockets and ICs for only 4K are provided. The expansion kit contains only the additional sockets and ICs to complete the 8K card.)

If you have been scared off building your own computer by stories of frighteningly close-spaced foil runs—fear not, for the H8 memory cards are an assembler's dream come true. Don't ask why, but the standard size solder pads for the inline DIP ICs used in the H8 seem very easy to solder. We had not one single solder bridge. We can attribute this to Heathkit's unusually thorough solder masking on the PC boards. Whatever, the PC assembly simply isn't difficult or "hazardous"; a major plus for the H8 as a hobbyist's project.

You Get What You Pay For. By now you have probably added up the cost of the basic computer, memory, I/O, etc., and realized you're fast approaching $1000. Actually, the very minimum, consisting of the computer, 8K of memory, a serial I/O-cassette interface, and a used KSR teletype terminal (which is available from some surplus dealers for about $300), will cost around $950. Add in another $210 if you want another 8K of memory (16K total). This is certainly not out of line with other "personal computers," and offers a heck of a lot more computing power in most instances.

For example, the H8 works in machine or assembly language, *or* BASIC. You can easily transfer control and entry back and forth between the terminal and panel monitor without bombing your program.

When running BASIC the panel readout keeps up continuous visual "chatter" about what's going on. You always know where you are in the memory (and Heathkit supplies a long list of BASIC routine locations). When loading from tape, the readout indicates the sequence of memory load and the top memory of the load; a value required if you want to make changes and then dump. When dumping a program the readout counts down to zero if the dump is clean.

Beep-Beep-Beep. Panel entries as well as dump and load functions are indicated by beeps from the internal speaker. A short or medium beep (depending on the particular entry) is sounded each time a panel key pad is pressed. If you enter a number or function and don't hear a beep you know you haven't pressed hard enough, or the computer didn't take the entry—do it again. A good load is signified by a single beep, as is the completion of a dump. A continuous string of beeps indicates a bad load: Repeat the load,

perhaps with the cassette's volume control set slightly higher, or check for a defective tape.

Brute Force Computing. The BASIC available from Heathkit has been updated several times. The *Benton Harbor BASIC* that is supplied with the computer kit is a floating point BASIC with the "transcendental math package." Its most serious limitation is lack of string variable manipulation. By itself, Bentor Harbor BASIC will meet the needs of almost all high school and general college students.

The *Extended Benton Harbor Basic* is in a class by itself when it comes to personal computers. It is a blockbuster of computing power. The most recent version, which was used for this month's Simply BASIC program, is at least equal to many of the best time-share BASICs and is better than most. It even permits a cheap cassette recorder to be used almost like a disk system.

This is not to imply there are no irritations. Heathkit BASICs have self-completion of several common commands and it cannot be turned off. If you enter PR on a terminal it will complete the command PRINT. Got any idea how many times we continued typing and got PRINT I, or INPUT U as the entry, to be rejected by the computer as an error? It's like being the sorceror's apprentice! We would prefer to do without self-completion of commands, or at least be able to turn it off.

Perhaps a more serious limitation is that an additional serial I/O card—with its attendant expense for an unneeded cassette interface—is required if you want to use a video terminal in conjunction with a TTY or serial printer. A separate parallel printer requires the even more expensive parallel I/O card. It would be convenient to have a single serial I/O card with two or more ports.

But these are really minor complaints in view of the fact the H8 system provides for extremely easy expansion, and Heathkit implies many accessories and peripherals will be forthcoming in the near future.

As it presently stands, the basic computer can accommodate up to four 8K memory cards, leaving one "motherboard slot" open for a connecting cable to an accessory cabinet housing more memory (up to the 8080A's limit) and peripheral controllers. Future memory cards may possibly utilize the most recent memory IC, which can provide 32K of memory in a single chip. All this is to come. The important fact is the H8 has been designed for easy expansion by the user.

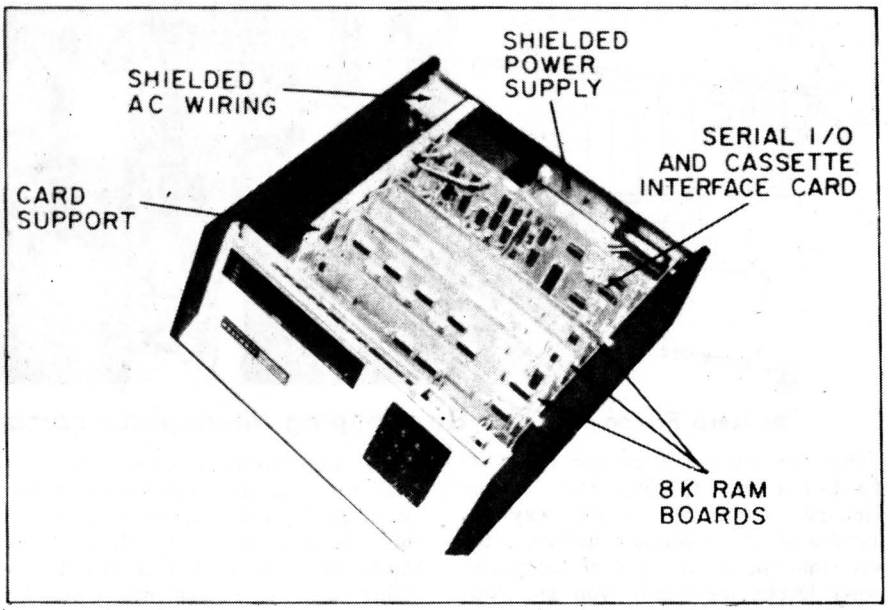
Is It For You? We have skipped over a long list of technical features because you can read about them in complete detail in the Heathkit catalog. As for total kit construction, we cover that in a future issue. Here we mentioned construction of the memory and interface cards, which is the most difficult part of assembly in terms of soldering—one solder bridge and your computer is inoperative.

Now we come to the *documentation* —a fancy word meaning instructions or operating manual. It is so extensive, and has been handled by so many different writers, you must have some knowledge of computing just to know where to look for what you want. Including the outstanding Heathkit BASIC Programming Course (EC-1100, $29.95), which is a *must have* if you want to get the most out of the H8 or any other computer using the BASIC language, the complete documentation runs in excess of four inches thick. For example, the basic instruction for connecting a teletype is in one book, the patches (memory data change) for providing the required two stop bits is in a second book, and the real patch for the latest version of basic turns out to be on an addenda sheet. Another inconvenience is the operating guides. If you leave them in the binder and flip to the opposite side you find they are upside down.

What's really needed for the rank beginner at personal computing is a "Big Dummy's Guide To The H8 Computer," with all the most commonly used commands and patches listed under clearly defined headings. With a Big Dummy's Guide a child coould get the H8 up and running in minutes, it's really tha. easy. But as it now stands, with its four-plus inches of documentation, better have someone around with a moderate knowledge of computers. (For example, while most of the world specifies the H8 type of readout as "octal," Heathkit calls it "offset octal." Got any idea how long it took us to figure out offset octal was plain, standard octal?)

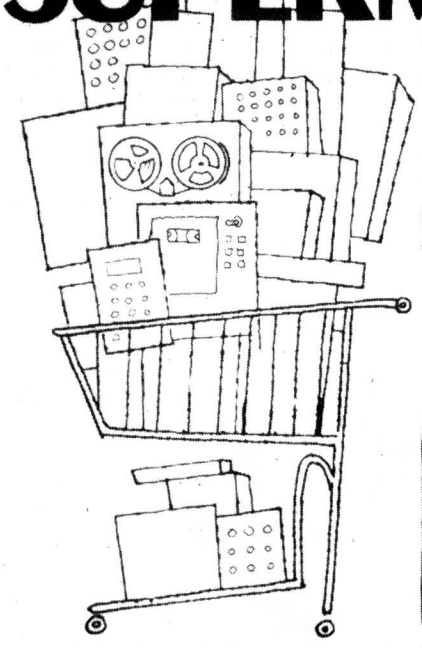
Summing Up. Minor complaints and the confusing documentation notwithstanding, the Heathkit H8 is an outstanding value as a personal computer for the student, starting way down at the seventh grade level. With Extended BASIC it is superb for junior and senior high school math. For the student seriously interested in advanced programming and technology it's probably the next best thing to having access to a good time-share system with other languages such as Fortran, PL/1, etc. (It's possible Heathkit is considering other languages in the future, particularly since Bell Laboratories, C-language is coming into vogue.)

The H8 does, of course, lend itself ideally to small business use, but as with all equipments reviewed in HOBBY COMPUTER HANDBOOK, we look at it in terms of a student and/or hobbyist. In those respects it's a winner all the way. Combined with it's "industrial grade" construction the H8 is an ideal choice for schools at any level.



You may have heard a lot about buss structures. The fact is that Heath supports theirs, and so do other manufacturers, so H8 expansion is as simple as plugging in boards.

COMPUTER SUPERMARKETS



by Herb Friedman

Go shopping where prime computers are your very best choice.

☐For the average electronic hobbyist the fastest way to get turned off to personal computing, or the way to spend a lot of unnecessary money, is to walk into the wrong type of computer store. Depending where you go, you might wind up more confused than when you walked in. The reason is that there are three distinctly different types of computer stores selling personal computing equipment, only one of which truly meets a hobbyist's needs.

The first type of personal computer store is essentially a retail parts dealer who caters to the advanced computer hobbyist—generally a college-level computer technology student designing or building a computer from scratch. The store stocks a wide assortment of computer theory and construction books as well as most commonly used solid-state devices, and special tools such as wire-wrap guns, pencils, boards, and sockets. The one thing these stores don't sell is hobby computer kits or accessories such as printers, recorders, etc. When they do have computer hardware it is generally industrial surplus which calls for considerable technical ability to get it operating in a hobby or personal computing system.

The second type of computer store caters primarily to the well-heeled customer. It wants to sell complete small systems including software (programs) to small businesses whose bookkeeping, billing and inventory systems are too small for a standard minicomputer but too large for a couple of office assistants. As a general rule, these stores handle some of the higher priced and higher quality equipments used by more advanced personal computer hobbyists; for example, many of them stock a CRT terminal that sells for about $1,000 which offers little advantage over a hobbyist-grade terminal that sells for around $500-$600. Another problem we've run across with these stores is that they simply don't understand the needs of the personal computer *hobbyist*. As an example, one local store sells the SWTP 6800 computer but virtually none of the accessories needed by the budget-conscious hobbyist.

Problems Solved. Finally, we come to the *real* personal computing store, the store with the staff and components for everyone—from the rank beginner to the small businessman. Note we have listed *staff* first, for without the correct staff an off-the-street customer will end up in a fog. Any decent personal computing store has on-staff (not a few hours a week) at least one person well versed in hardware—their use, interconnection, applications and, most important, *debugging* (someone might have to find your mistakes in a kit), and a specialist in software (programs). Both might be the same person, but there must be someone who can handle software, hardware, and debugging problems. Often, college-level technology students are the specialists.

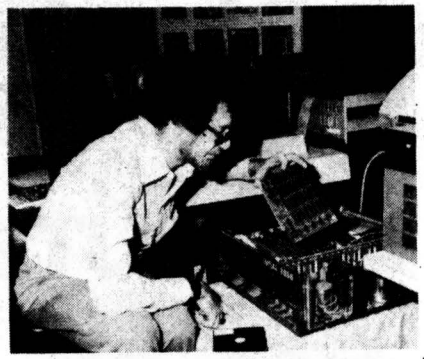
A look at the Computer Mart of New York is a good example of what a personal computing store, catering to electronic hobbyists, should be like. The pictures shown were taken at the Computer Mart.

Firstly, the store must handle a broad line of equipments, starting at the budget end. For example, the Computer Mart handles basic computer kits that can be expanded by the builder as needed or as the budget will allow. They have wired systems complete with a resident high level language (usually BASIC); and minicomputer systems suitable for small businesses. Similiarly, a good computer store for personal computer hobbyists can supply peripherals ranging from budget priced terminal kits to color CRT terminals.

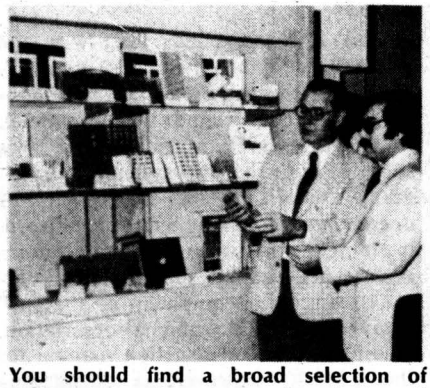
Most important, a true experimenter's store should be able to start you off inexpensively. For example, you might not be able to afford, or even need in the future, a recording system (for recording or feeding programs), but how will you initially load a language such as BASIC into your computer? A good hobby computer shop will tell you about an inexpensive teletype paper-tape reader where you pull the tape through by hand. There's no fancy mo-

Here a couple of strav customers—which include HOBBY COMPUTER HANDBOOK'S Editor-in-Chief Julian Martin (right)—checking out the display of software, magazines.

Just killing time? Try a game of Blackjack on a Compucolor Computer. The cards are dealt in full-color along with wisecracks from the computer-dealer.



What to do if you build your own and have trouble? At Computer Mart, one of the store's technical people checks out a memory board in a working computer.



You should find a broad selection of accessories and prototype boards for all computers stocked. Our two customers look over the latest wirewrap boards.

If you need a demonstration it should be very easy to get any system in a computer store up and running. Here Julian and the stores technician check out the BASIC in a SOL computer. Almost every terminal sold can be used by the prospective buyer for purposes of comparing the features and the "feel" of varying brands.

tor drive but there's no fancy price, either, and you can load TTY paper tape programs at budget prices.

A store such as the Computer Mart will provide for extensive hands-on experience. Several different systems will be arranged so they can all be in operation at the same time, allowing several hobbyists or groups to get the "feel" of different computers and peripherals. At the Computer Mart they have an island with several complex systems, and individual stand-alone computers arranged out of the way on a shelf. One or two users don't tie up all the demonstration equipments.

Accessories, such as program tapes and cassettes, small devices such as a gadget that converts video output from a terminal to a RF signal that can be fed into a TV receiver, or a hand-operated paper tape reader, will usually be found in a glass display case. Unlike the stores that are primarily interested in complete systems for business installations a good personal computer store will usually stock or handle a wide range of accessories for each of the computers they sell even when not made by the computer manufacturer. For example, at the time we made the pictures at the Computer Mart, they sold the complete SWTP computer line and the Smoke Signal Broadcasting disc system for the SWTP computer. Similarly, the store sold Seals memories for several types and brands of computers, as well as development/circuit boards for the SWTP computer and computers using the S-100 bus (Imsai, Altair, etc.).

All the Extras. A good hobbyist store should have an extensive selection of books and magazines specifically intended for the beginner, and the reading material should accommodate *what the customer knows he wants,* not what the owner of the store believes the customer should have. For example, there are the computer hardware *nuts* who want to know the ins and outs of every electron flowing in the system. Fine, there are many books and magazines for the hardware specialist. On the flip side of the coin are the *programmers* who don't give two hoots how a computer works, they just want to be able to press a button and have a system come to life. The programmers are the end users and they need lots of books on programming and programs. A good selection of programmer oriented books can be the most difficult thing to come by. Often, the selection represents the store owner's level—too basic, or too advanced. A good experimenter's store should handle both ends, and everything in-between, with a thorough selection of the personal computing magazines, not just the two with the largest circulation.

Next, we come to specialized tools and support hardware. Resistors, capacitors and even most solid-state devices used in computers can be obtained from electronic parts distributors. But, what if all you want is one mini-floppy disc, which sells for about $5? Standard pack is 5-for-$25. A good personal computing store should be willing to break the pack and sell you one. Need data cassettes (a higher quality Philips-type cassette)? Again, a good hobbyist store will have them in stock or get them for you on short notice. Has the computer manufacturer updated his BASIC program (higher language)? A store like the Computer Mart will have its order in just as soon as it's announced.

Building a computer kit? There's no reason you have to go to an electronics parts distributor for sockets and connectors. A decent computer shop stocks sockets and connectors for all the heavy equipment, and provides some sort of technical service. Many personal computer stores will check out a kit board purchased from them at no charge by plugging it into a working computer. Many stores also provide extensive repair service, usually at charges representing the time spent undoing your soldering errors. As a rule of thumb, unless you have a lot of backup support

(Continued on page 80)

HCH checks out the...

RADIO SHACK TRS-80 COMPUTER

This system brings the Computer Age into your livingroom.

☐ Though there has been much talk of personal computers in the sense of two cars in every garage and a computer in the den, until the Radio Shack-TRS-80 there was nothing that could be used by someone totally unfamiliar with electronics. If you couldn't tell a Molex socket from a D-connector there wasn't much chance you'd be able to *interface* (fancy term for interconnecting) the components that go into making up a basic computer system.

The TRS-80, however, is a whole different idea in personal computing. Just push together three common DIN connectors such as you might find in a typical hi-fi system and you have an instant computer system pre-programmed with BASIC. Press the power switch and your computer system is up and running.

The TRS-80 consists of three independent equipments. First, there's the computer itself with it's associated plug-in power supply. Housed in a cabinet approximately 16½-in. wide x 3⅝-in. high x 8-in. deep, with an integral keyboard, the computer has a resident 4K basic, graphics, and 4K RAM (random access memory) which can be expanded to 16K on-board, or 62K with a planned outboard accessory. The back of the computer has DIN connectors for it's power supply, the video output (for CRT display), and a tape recorder. The main power on-off switch is located adjacent to the connectors. On the rear left is a cover plate for an *expansion port* (fancy term for printed circuit connector), and the *reset switch* that bails you out if your program *bombs* and locks the computer into a loop.

The basic computer and its plug-in power supply is priced at $399.95 (assuming it ever becomes available as a separate item).

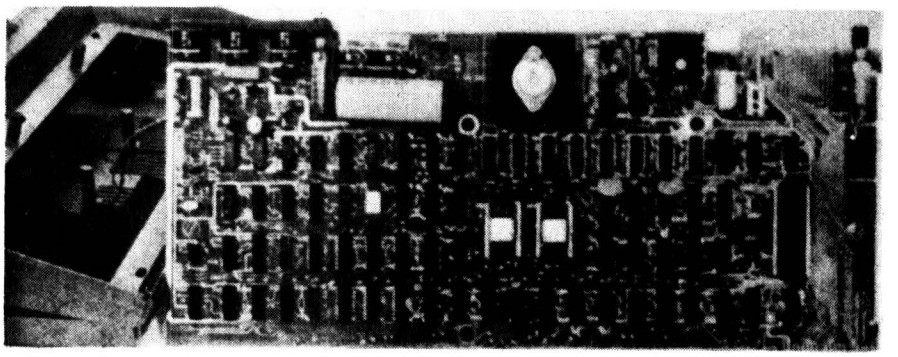
The second equipment in the system is a 12-inch video display monitor priced at $199.95. Front panel operator controls are the on-off switch, and brightness and contrast adjustments. Unlike the usual video monitor which has a UHF or BNC connector input—and you provide the matching cable and connectors—the Radio Shack video display has a permanently attached input cable with a DIN connector that plugs into the computer.

The final equipment in the package is a Realistic CTR-41 cassette recorder usually priced at $49.95. It is supplied with signal and control cables that plug directly into the computer. The recorder is used for storing user-programs, or for loading the computer with a program from several program packages available from Radio Shack—among them a payroll program for up to twelve employees, a personal finance program, and a special math education program.

Though the three items individually represent a cost of $649.95, the package price, including a 300 page instruction / programming manual, is $599.95.

How It Works. Let's start with the manual. The final version was not available when this article was prepared. Instead, an abbreviated user's manual, sufficient to get anyone started, was supplied with the first systems. It is literally a work of art. The quality of *documentation* (another fancy word, meaning instructions and technical information) usually supplied with personal computing equipment is "the pits" —*abysmal* is too good a description. The preliminary instruction manual for the TRS-80, however, is outstanding by any consumer standards. We can only hope the same person prepared the final 300 page manual. The only problem with the manual is that the demonstration "multiplication tables" program has two typographical errors and the program won't run. This is a fun program and will probably be the one you'll try at a Radio Shack store so make the following changes: statement #70 should read ". . . . . GOTO 130"; statement #110 should read ". . . . . GOTO 20". (As we said, this is a preliminary manual.)

To use the computer, just press the power switch. If the screen fills with scrambled alpha-numerics simply press the power switch again and you'll see the word READY. The computer is



This is the actual computer, just one IC after another. The two large chips in the middle, with the white centers, are plug-in ROMs (Read Only Memories). These are the chips which contain BASIC.

TRS-80

now ready for use.

If you want to save a program you need only install a good quality audio cassette in the recorder, preset the recorder's controls to record, and enter CSAVE on the terminal. The computer automatically starts the recorder and feeds the program to the tape. Taped programs can be loaded into the computer by entering CLOAD on the terminal. (The computer automatically converts the digital pulses to audio signals for recording and vice versa for loading.)

Radio Shack has announced intentions to make available, at a later time, peripherals for "hard copy" (a printer) and a disc system. By the time you are reading this one or more accessories might already be available.

Though the overall concept for the TRS-80 personal computer is good, the 4K BASIC leaves a bit to be desired. It somewhat limits the potential, primary consumer use of a computer—that of an educational aid. The 4K BASIC lacks trignometric and transcendental functions, limiting it's math level to elementary grades. Junior High and High School would require, at the very least, the trignometric functions. Radio Shack has announced a Level-II BASIC will be made available—which we assume will be essentially some form of 8K BASIC with the trig and transcendental functions.

The BASIC is in plug-in ROM (read only memories) so we assume Level-II BASIC would be installed by simply exchanging one or two ICs. (The computer easily comes apart by removing a few screws.)

The recorder runs at 300 baud, so saving and loading a user program can take considerable time. One of our programs took almost 8 minutes to save, and 8 minutes to load. Radio Shack undoubtedly used 300 baud in order to utilize an inexpensive audio recorder, but a high speed digital recorder would be preferable. There are indications from Radio Shack that they are planning for high speed mass-data recorders in the near future.

Why For Me? Perhaps you're impressed by the specifications of a computer such as the TRS-80, but you're still hesitant about letting the "computer age" right into your living-room. You may be wondering: isn't a home computer mostly for an engineer or accountant?

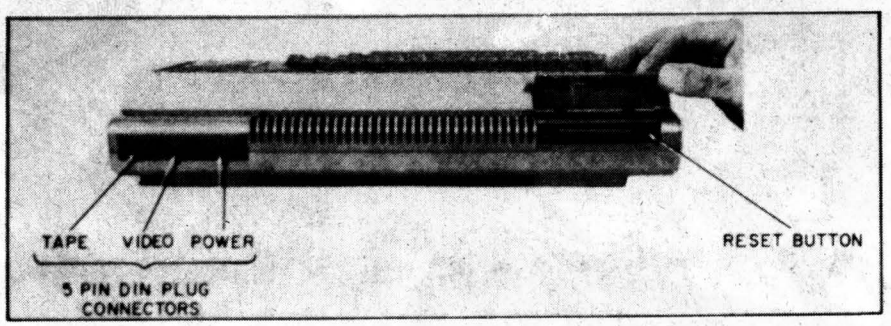
Right now, the field of Hobby Computing is expanding at a truly amazing rate and encompasses people from every walk of life and of a myriad of varying interests. A typical computer user can be a High-School student running a few amazingly realistic games in his spare time; a housewife using her own computer system to help plan a big dinner from recipes designed for a much smaller crowd; to a small businessman doing the payroll and invoicing, which used to take his secretary days, in just a matter of minutes.

Computers are no longer in the future. Computers are here, right now, ready to change your life for the better.

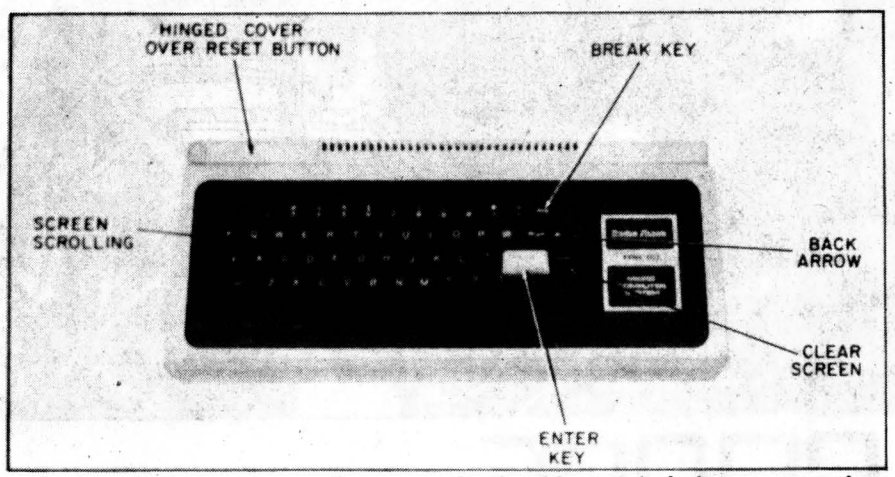
Radio Shack has recognized the needs of the home computerist and has supplied taped programs which will help you get started on using the TRS-80 on a day-to-day basis. The "Games" tape which comes with the TRS-80 introduces you to the fun aspects of computing. More tapes are available from Radio Shack which include such things as programs to help you run your kitchen and housekeeping more economically and efficiently; tapes that program the computer to be your private, business secretary ready at any time to do your books and payroll perhaps more accurately—and certainly quicker —than ever before.

Once you get started, you won't be able to stop. Courses, such as our introduction to the BASIC Language which is found elsewhere in this magazine, will enable you to plan your own programs and to solve your own, special problems without having to depend on taped cassettes. In fact, one of the most exciting aspects of this new hobby is the programming. It is a real thrill to set yourself and your computer the goal of solving some problem or other, to program the computer, and watch that problem melt away.

For the past few years, thousands of people have been introduced to the Hobby Computing field. With the introduction of the TRS-80 (and other computers like it) into the marketplace there will soon be millions of people, just like yourself, using computers in their everyday lives. ■



The TRS-80 microcomputer from Radio Shack can be the first step in bringing the Computer Age into your own home! It's available in a complete system, fully assembled, and you can be programming in just a few hours after you take a TRS-80 home. Already, thousands of people are involved in the new Personal Computing hobby. A system such as the TRS-80 can be a fantastic introduction to it.



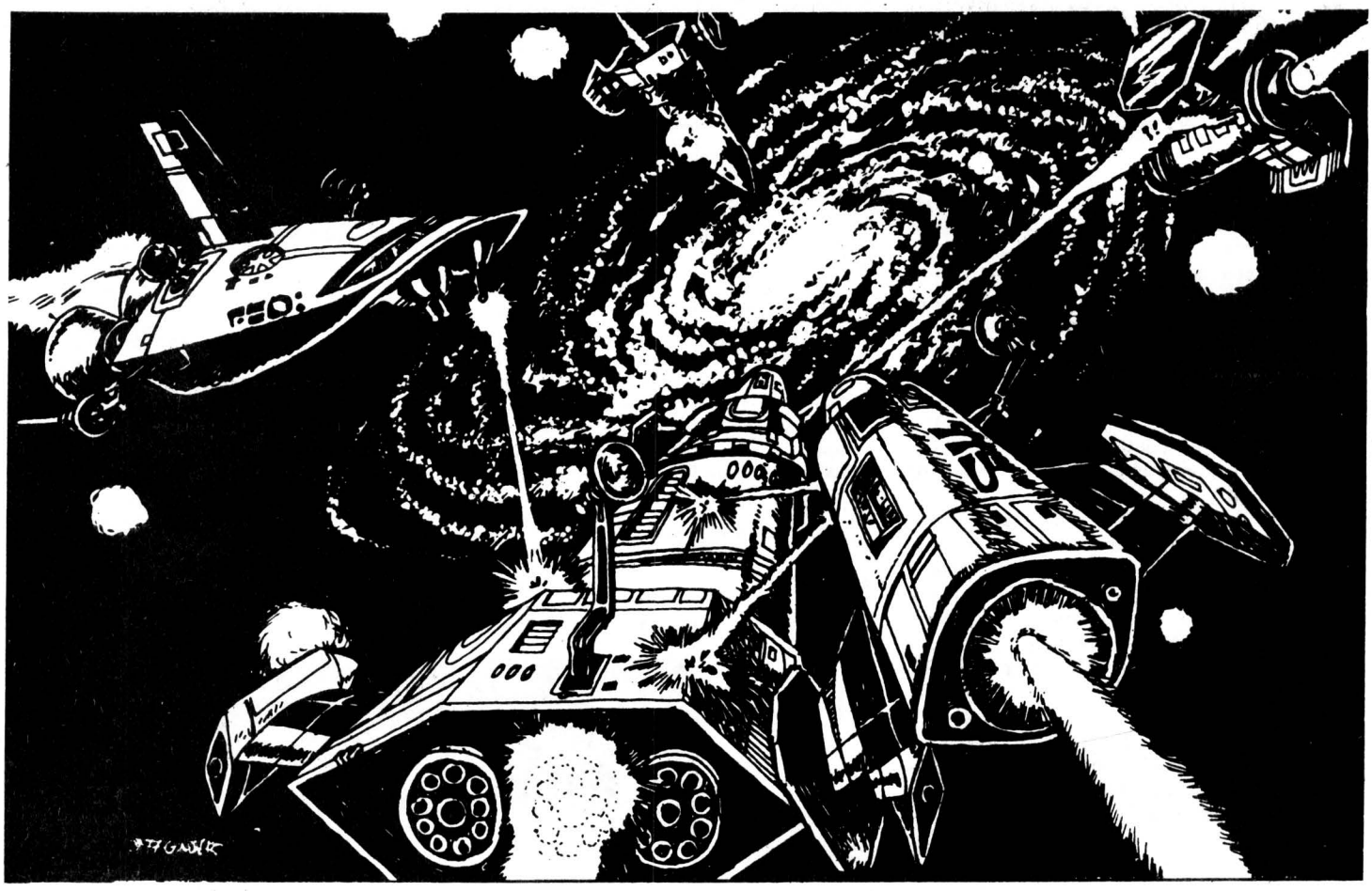
The three DIN connectors on the rear are for the video terminal, the power supply, and an audio cassette recorder. A cover, shown lifted, conceals an expansion port which will be used for optional accessories in the future—things such as magnetic and disc recorders, hard-copy printers and a variety of memory expansion boards.

HOBBY COMPUTER HANDBOOK

HOBBY COMPUTER HANDBOOK 4
YOU AND YOUR COMPUTER 8
CLAUDIA'S COMPUTERS 12
COMPUTERS PHONE THE FUTURE 13
THE MICROPROCESSOR WORLD 18
SWTP 6800 COMPUTER 20
HEATHKIT H8 HOME COMPUTER 23
COMPUTER SUPERMARKETS 26
RADIO SHACK TRS-80 COMPUTER 28
SPACE: 30
COSMAC VIP COMPUTER GAME 32
PENNYWHISTLE 103 ORIGINATE MODEM 36
MICROPROCESSORS AT THE MIKE 38
TEXAS INSTRUMENTS 58 PROGRAMMABLE CALCULATOR 40
DATAC 1000T COMPUTER 42
APPLE 11 COMPUTER 44
BIO LOGIC 46
CHESSBOARD RUMBLE 49
MICROPROCESSOR NUMBERING SYSTEMS 51
PROGRAMMING WITH BASIC 65
HEARING AGAIN THROUGH MICROPROCESSORS 79

This publication was originally produced in the U.S.A. Accordingly, prices referred to are those pertaining in the U.S. at the time.

Published by Page Publications Pty. Ltd. and printed by Rotary Colorprint Co. Pty. Ltd.
Both of Butt and Clisdell Streets, Surry Hills, N.S.W. 2010.
Distributed by Gordon & Gotch (A/asia) Ltd., Melbourne-Sydney.
* Recommended and maximum price only.



SPACE:

Run a farflung stellar Empire on pennies a day!

By Neil Shapiro WB2KQI

Look up past the treetops, further still, beyond the clouds and the familiar blue sky into the black velvet of Space. Height no longer has meaning, and the stars are sprinkled like a diamond cutter's dust and scattered to the farmost edges of Infinity. In the silence of that void, like technological phantoms, interstellar ships of competing galactic empires move at unimaginable speeds through uncharted dimensions. The ships fly on their missions of power and conquest, linking together the blue and habitable worlds which circle yellow stars, and often these ships clash terribly in that night, temporarily bringing light to an emptiness never lit before.

In a way, all of this is fantasy—but on the other hand, it seems quite real to the hundreds of people who play GALAXY II each month by mail. Once every turn a prospective Galactic Emperor must make hard and fast decisions on how to run his own segment of the galaxy. Everything, from building ships and naming their captains to founding colonies, fighting battles, collecting and disbursing taxes, and much more, is all figured into each turn. There are more than forty players in each game who send in monthly moves. The moves are then collated, followed through, and any player interactions decided, all by computer.

Of course, it's not just any computer which can handle the complexity of a game such as GALAXY II. The game is run on two System 370 model 168's from IBM. These are huge computers with over seven megabytes of real core storage. In fact, these computers use something termed "virtual storage" which, as the name implies, means that the available memory space is virtually unlimited depending on how the system is configured.

Does this mean that only computer programmers and high-level scientists are among those playing GALAXY II? Well, many of the players (all of whom hope to become a Galactic Emperor) come from backgrounds widely divorced from the sciences and research. Brett Tondreau, GALAXY II's designer, states, "We're looking for overworked, highly educated people who are jaded with the cursory forms of entertainment currently so common in the TV and movie theatres. We need people who are conscious of what they like and have a very high respect for both their own creative energies and those of others."

So, if you're tired of watching *Star Trek* re-runs on the tube, here's your chance to set out on your own mission and go 'where no man has gone before.' All it takes is the creative energy to be able to envision what it would be like to run your own galactic empire, and the desire to give it a try. The rules are complex in their entirety, but each part of the whole can be readily enough comprehended with just a little effort. Once you know what you're doing, you would be hard put to swear that all those spaceships, planets and colonies *don't* exist. It is simply amazing how receiving personal reports of a mission accomplished from one of your favor-

SPACE:

ite starship captains tends to make you believe in everything that's going on.

Because of the wealth of detail involved in GALAXY II, there is really no such thing as a typical turn. There are, however, typical parts to most turns, operations which are done again and again but put together in varying ways for different goals.

For instance, let's say you're ready to launch a new spaceship. Your first act is to see that the planet launching that ship has enough Matter-Antimatter-Drive units (MADs) to contribute to a vessel of the size you plan on designing. If not, you'll have to decide how much of that planet's industrial capabilities to assign to producing those MADs, and then you'll have to begin juggling priorities. To make it easy, we will assume that the planet has already stockpiled 8 MADs, which is enough to fashion a Lance-Class vessel.

So, you have the eight MADs. Now, each MAD has to be contained within a hull unit (HU). You will then have to assign at least 8 HUs to the ship. Again, you have to decide to either build new HUs or to use any stockpiled.

Let's not forget the fact that the ship has to be crewed. You may want to also assign it a goodly number of colonists in "cold sleep," along with maybe one or two high-level passengers. All of these people breathe, and so you will have to assign a value of Life Support (LS) that will last at least as long as the mission you plan. To get those peopeople where you want to send them will require you to assign Power Support (PS).

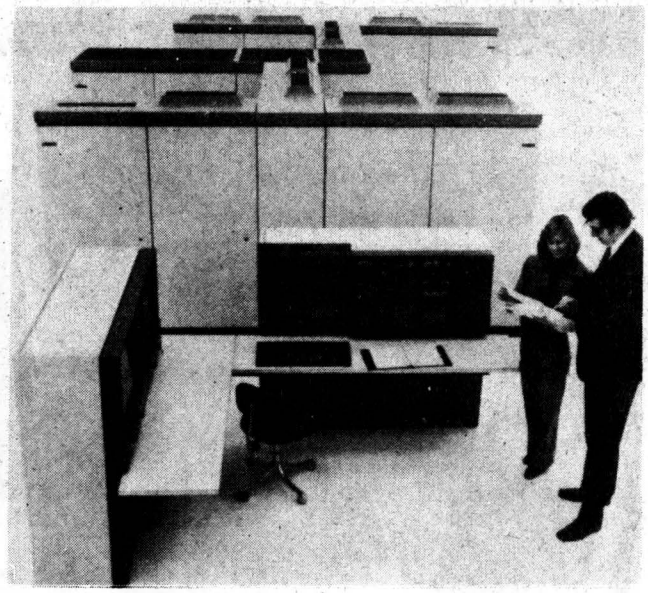
Unfortunately (though it's a lot of fun!), everyone "out there" isn't going to be friendly. It would be a good idea to assign that Lance-class ship a certain number of Beams (BEs) for offense and a goodly number of Shields (SHs) for defense. Combat will be decided using these variables along with how experienced your ship's captain is from previous battles or missions.

Further, you may want to assign each ship a number of Probes (PBs) so that it may gather intelligence for you. Of course, don't forget the Screens (SCs) which can shield that ship from an enemy's probe and mask its identity.

Figure on having only a small fleet in the beginning of the game. Sooner or later, once you figure out which stars hide your opponents' own degenerate races as opposed to your own race of superbeings, you'll have to build a bigger fleet to root the blighters out.

Be sure you don't forget your planetary development. This is one of the

(Continued on page 82)



GALAXY II is run on... an IBM-370 model 168 like the one shown here. The computer offers seven megabytes of real core storage along with virtual storage systems which give an almost unlimited storage capability. The game makes use of the system's peripherals which are configured to include such things as: 32 disk drives, three banks of tape drives and IBM's fastest printer terminal. Right now, GALAXY II is using only 500K of memory so there is plenty of room for future Empires to expand and war into.

```
 WHAT FOLLOWS IS AN ACCOUNT OF THE ACTIONS ON THE PLANET AVALON   GALACTIC COORDINATES 14-14-3 IN THE POST FLIGHT PHASE
COLONY   LOCATION GOVERNOR   BEAMS SHIELDS  PROBES SCREENS    MADS   HULLS  ECONOMIC  MINERAL-II  INDUSTRIAL UNITS   CAPACITY
AVALON    14-14-3 GAWAIN        18      20      20      20       1       0  RESOURCES        466             2817          13.32
                EXPERIENCE                    DESIGNATION                    MINERAL-I MINERAL-III  UNIVERSAL UNITS  EFFICIENCY
                         1                         AH                          522         293          -21988          3.97
```

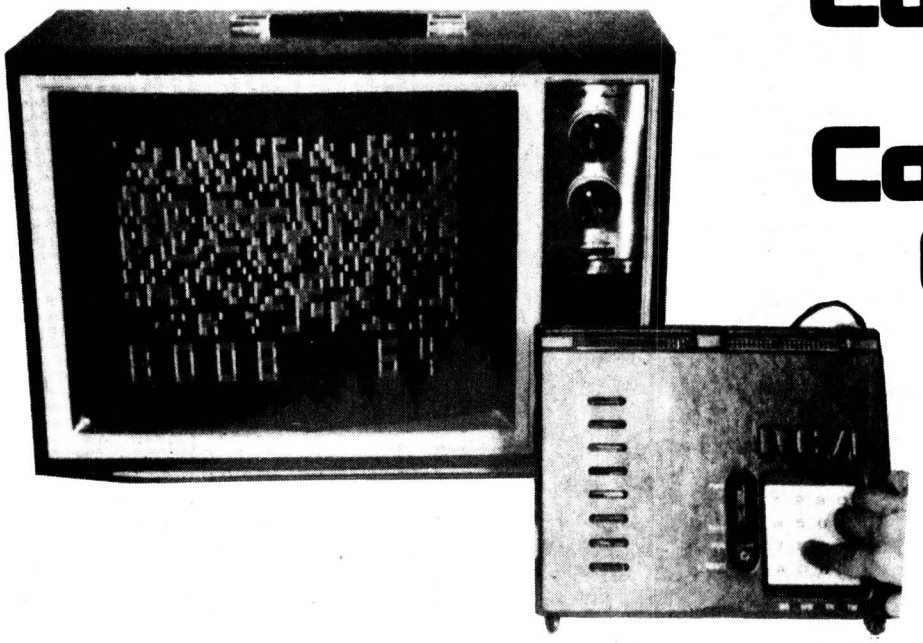
Each month the GALAXY II computer tells you how your dreams of Empire are being made into reality—either that or how you're gradually being wiped from the map by your opponents. Here we see a typical report on a planet by the name of Avalon, which is governed by a gentleman named Gawain. The planet is in fairly good shape mineral-wise, and is just beginning to show signs of economic expansion. Obviously the treasury has been depleted, as shown by the -21988 Universal Unit figure. Hopefully, this money has been invested soundly and new factories (Industrial Units) have been planned as well as investing in the indexes of Capacity and Efficiency. If this plan of deficit financing succeeds then Gawain will wind up governor of a rip-roaring planet.

```
SHIPS LOG OF THE VESSEL IGRAYNE   CLASS: MSH        FIRST FLIGHT PHASE

FLIGHT EXECUTED.  PRESENT LOCATION OF IGRAYNE  IS 14-14-2.
CAPTAIN BLAMOR   REPORTING A SUCCESSFUL FOUNDATION OF COLONY CORNWALL AT 14-14-2 WITH   30 INDUSTRIAL UNITS.
COLONY CORNWALL REPORTS SUCCESSFUL ADAPTATION.  INITIAL CONDITIONS INDICATE THE CAPACITY OF PLANET AS 0.33 . -CAPT. BLAMOR
WE HAVE TRANSFERRED        12 SCS TO THE COLONY CORNWALL
VESSEL   LOCATION  CAPTAIN   BEAMS SHIELDS  PROBES SCREENS    MADS   SECURITY    TRIBUTE  POWER SUPPORT  MINERAL CARGO  MINERAL-II
IGRAYNE   14-14-2  BLAMOR       20      20       0       0       6      (S)           0       25.1999                        0
                EXPERIENCE  PASSENGER & EXPERIENCE  CLASS  HULLS  DESIGNATION  INDUNITS  LIFE SUPPORT      MINERAL-I  MINERAL-III
                        20                          C   MSH       6      AH           0       79.9951              0            0

SOLAR SYSTEM REPORT AT 14-14 (RACE,DG,CONO.,COLONY): (NO RACE  UH   UNNAMED ) (LOGRES   AH   CORNWALL) (LOGRES   AH   AVALON  )
AND  2 VESSELS (BY (RB,HULL,DG,C,RACE): ORBIT 0: ORBIT 1: ORBIT 2: IGRAYNE  AH  MSH  LOGRES   ORBIT 3: YSOLDE   AH  MSL  LOGRES
```

The computer also reports on each and every one of your starships; tells you where they've been, what they've seen and what actions they've taken. Here we see the log of the Igrayne, under the command of the noble Captain Blamor. The good Captain is able to report to his Leader that the colony of Cornwall was successfully founded and the colonists transported safely there. His ship is now empty of cargo (INDUNITS) as all the colonists have been dropped off. The Igrayne's armament, while not too very formidable, consists of a good selection of beams, shields, screens and MAD. The Igrayne under Blamor's command would be a tough nut to crack

HCH assembles the...

Cosmac VIP Computer Game



Enjoy games and graphics too with this super RCA kit

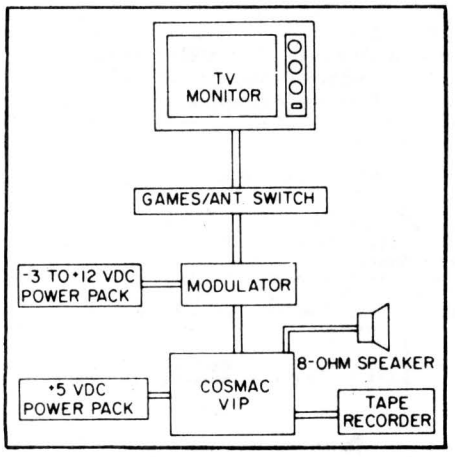
RCA'S NEW "FUN MACHINE" is a home hobby computer that offers remarkable entertainment and educational potentials at modest cost. Assemble this one-card COSMAC VIP (Video Interface Processor) kit, feed the output to a conventional black-and-white TV set through a simple modulator, add a small cassette tape recorder for permanently storing programs, and settle down to virtually endless experimentation.

You can immediately program and play 20 different games *without* spending another cent on program software because the games are included in the $249 purchase price of the VIP kit. Right there is where you can save a bundle when you figure that many other game computers on the market require additional investments of twenty dollars and more for each game or two you want to play. Your only cost is for blank cassettes on which to write your own game programs. If you can't locate a COSMAC VIP in a local electronics or computer store, you can write to RCA Solid State Division, Box 3200, Somerville, NJ 08876 and circle No. 48 on the Reader's Service Coupon.

More Than Games. In addition to playing the games and creating computer art and animation on your TV screen, you can go on to develop your own programs utilizing a special, "easy-to-learn" CHIP-8 hexadecimal language. And if you have a mind to go deeper, you can obtain hands-on experience using machine language.

The COSMAC VIP is *not* for you if you seek a machine that can process income tax returns and file cooking recipes. It quite possibly *is* for you if fun experimentation is the desired objective. You could experiment endlessly without adding another item to the system. However, hardware hackers will appreciate the complete external interface capabilities that permit systems expansion for such additional applications as model railroad control, music synthesis and even color graphics.

Memory can be increased to 4K bytes by just adding four ICs to the printed circuit card. This permits sophisticated programs for a system of this class. A comprehensive 44 line interface also provided can be used to add "almost anything" to the system including 32K bytes of programmable memory; this interface provides all the COSMAC microprocessor signals. Those wanting to add existing IO devices such as relays, music synthesizers, printers, or an ASCII keyboard, can use a parallel IO port option that is provided. Adding four readily available ICs to the printed circuit card provides an 8 bit output port, an 8 bit input port, and handshaking signals on 22 interface pads.



This block diagram shows the interconnect arrangement of the COSMAC VIP and its various inputs, outputs and power supplys.

Be advised that much of the documentation provided with the COSMAC VIP, especially material intended for *extended* experimentation, is not written for newcomers to the hobby computer field. In fact, making the basic system operational will be much easier for those who already have some knowledge about computing terms and procedures such as RAM, ROM, memory addressing, instructions, bytes and the like. If the VIP is to be your first venture into hobby computing, you can pick up the needed background knowledge through a little extra reading. Later on, I'll provide some tips and suggestions to get you over some of the rougher spots I had to labor through.

The COSMAC VIP utilizes RCA's 91-instruction CDP1802 microproces-

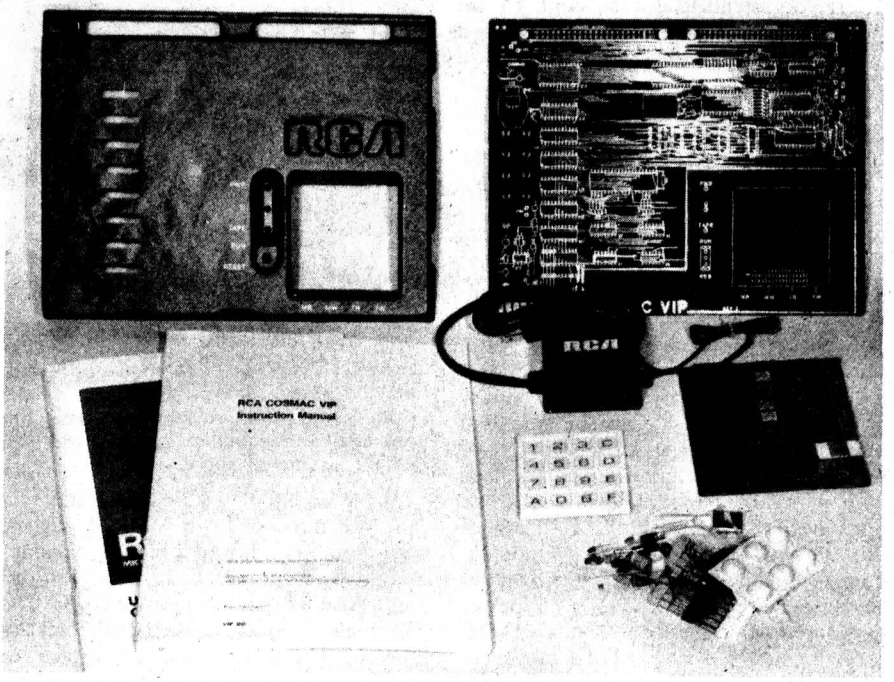
Cosmac VIP

sor and a graphic video display interface. Other features include: a 2048-byte RAM (Random Access Memory); touch-pad type of hexadecimal keyboard; 100-byte-per-second audio cassette interface; wall-pack type power supply; sound circuits used for signal tones and games; 512-byte ROM (Read Only Memory); capability for on-card RAM expansion up to 4096 bytes; on-card parallel I/O port expansion capability.

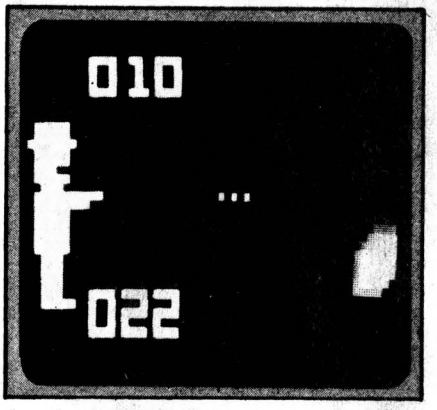
Assembly. You'll have no problems putting the kit together provided you have some experience in kit building. Just be sure to observe the cautions when handling static-sensitive IC's, and check the polarity of your power source before turning on the juice to eliminate the chance of damaging some ICs. Plug-in sockets are provided for only some of the ICs. I recommend that you pick up a few more from your local electronics retailer for the other ICs; unsoldering a defective or improperly oriented IC can be nerve-wracking. The VIP comes with a plastic cover that protects the components on the card, but for some reason no *bottom* protective panel is provided. I strongly urge that you add a protective bottom, perhaps of Masonite. Make sure to leave the rubber feet on the card to provide space for ventilating air flow.

You'll need a modulator if you want to use a conventional black-and-white TV set as a video readout; the VIP can be connected directly to a commercial computer video display. The modulator is *not* supplied by RCA, so you will have to shop around for a suitable unit to convert the VIP's video signal to an RF signal that can be fed to the antenna terminals of a TV set.

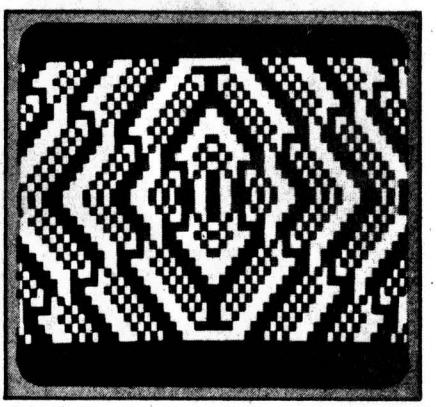
I tried a *Pixe-Verter* PXV-2A modulator (ATV Research, 12 & Broadway, Dakota City, Neb. 68731) and at $8.50 it seems to be a good value. The kit goes together quite easily if you are able to decipher the rather crude components placement drawings. Just be sure that you put the components on the plain side of the PC board. This converter has two controls: one is for tuning the output to an unused TV channel (channel 3 in most areas); the other very handy control (not found on other makes of modulator) is used to eliminate bleeding whites in the video display that might be caused by overmodulation. If tuning to channel 3 seems difficult, try moving the "channel select tap" jumper wire to another section of the coil on the PC board, then retuning again to channel 3.



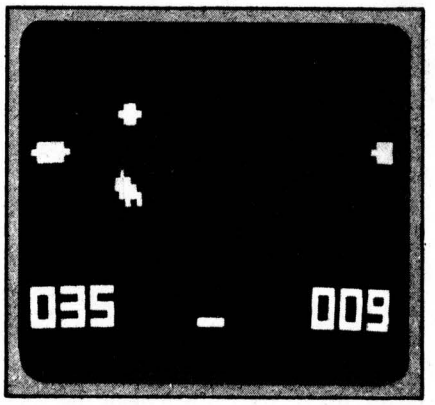
Assembly of the COSMAC VIP kit takes only from 3 to 4 hours because of the small number of components that need to be assembled. The kit includes a 5-VDC power pack, instruction manual, and a quite-technical manual for the CDP 1802 COSMAC Microprocessor which need not be read to build and use the VIP.



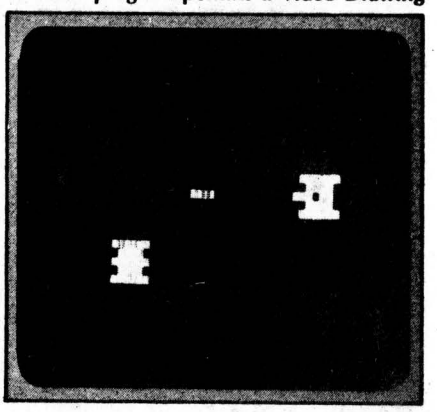
Shooting at a Moving Target—shoot straight, or angled up or down by hitting keys 3, 6 or 9 respectively. Target at right moves in a random manner in right sector.



Keleidoscope—not really a game but a program to generate an almost endless pattern of geometric shapes—the patterns evolve in stages rather than just appearing. Another program permits a Video Drawing



Space Intercept—Blast UFOs out of the sky by adjusting the angle-of-ascent of the rockets. Stop motion photography in this photo creates the triple image.

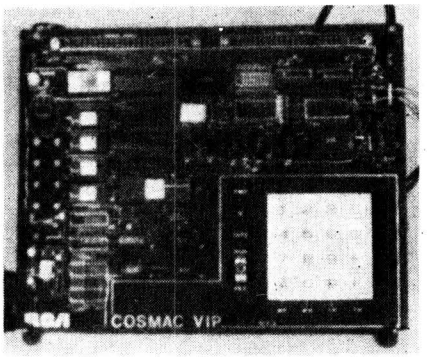


Armored Vehicle Clash—Try to blast the elusive tank that is under the computer's control. Key "F" fires the gun.

If you intend to use the same TV set for normal broadcast viewing, you need a games/antenna switch with at least a 60 dB rating and made especially for this purpose. This goes onto your regular antenna terminals on the TV set, and the TV roof antenna lead and the VIP modulator output go to indicated terminals on the switch. This switch is not part of the Pixe-Verter package, so shop for it locally.

Use coax cable (RG-59/U or equivalent) to connect the modulator to the TV set and to the VIP. A 72/300 ohm transformer at the TV set is not needed.

You can use a 6-volt lantern battery to power the Pixe-Verter; be sure that the *negative* side goes to the modulator input and positive goes to ground. Two other brands of modulators I tested use *positive* DC, so watch the connections if you switch from one type of modulator to another in the course of your experimentation. You can also power the modulator with a regulated DC voltage supply such as a Heathkit IP-2728, or even with an inexpensive AC adapter. For some reason, my system works better at a somewhat reduced voltage of about −3.5 VDC instead of the recommended −6 VDC. I suggest that you add a pot to a battery or adapter type power supply to vary the voltage. Also add an inexpensive 0-15 VDC voltmeter if your power pack lacks a meter. There's another advantage to adding a voltage adjusting pot; slight adjustments of the video display at times can be made much more easily with a voltage change than by digging into the modulator to retune.



Eighteen ICs and other components, including the hexadecimal keyboard, go on the card. Sockets are provided for some, but not all ICs. The author suggests that you obtain additional sockets.

The Pixe-Verter, or any other modulator you choose, will work only on a properly functioning TV set. Even a set that puts out a good TV picture may not perform as well with the weaker RF signal from the computer. It may be necessary to slightly readjust the vertical height and vertical linearity controls of the set, as well as fine tuning and horizontal hold.

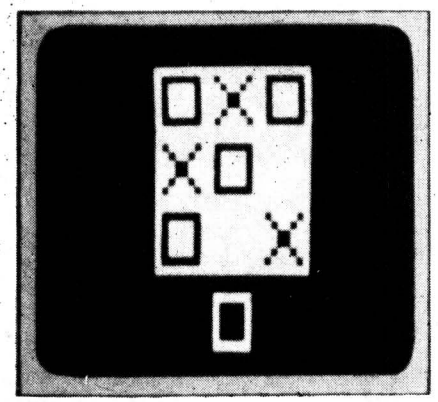
Sup-'R'-Mod-II, a black-and-white modulator sold by M & R Enterprises (PO box 61011, Sunnyvale, CA 94088), costs $25.95 because it comes fully assembled, with a connecting cable and a deluxe games/antenna switch. This unit utilizes a +12 VDC power supply (not provided), and has a single control for tuning to channel 3.

The first M & R unit I received was defective in several respects (the assembler must have had a bad morning), but the company promptly sent a second unit when I complained. The replacement unit also came with four pages of instructions, instead of the one page provided with the defective unit.

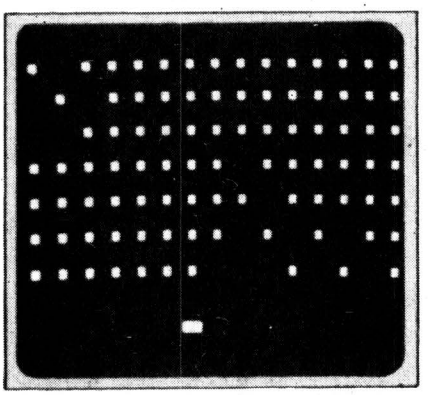
My main problem with the M & R modulator was getting adequate picture contrast, and in wholly eliminating the slightly annoying moiré type interference patterns in the whites of the display image. I snapped up the contrast to a wholly acceptable level by adding a 4700-ohm resistor in parallel with the 10K resistor next to the shielded box on the PC board. I never did get rid of all the moiré interference, and fine tuning was a bit more difficult, I thought, than with the Pixe-Verter. The main advantage of the M & R unit is that it comes fully assembled and provides a good antenna switch and connecting cord. Also, it might work much better with your more modern transistorized TV set than with my old vacuum tube job.

When everything is connected and working properly, and you turn on the VIP as instructed in the manual, you should see a random pattern made up of square white spots on a completely black background. When you key 8008 on the hex keyboard, this number should appear in the lower left corner and number 64 should appear in the lower right corner of the display. If you get all that, you are in business!

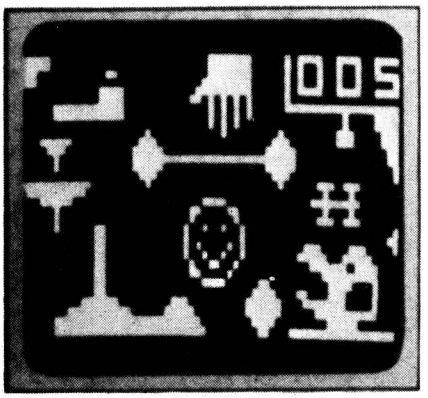
Hexadecimal System. The COSMAC VIP utilizes hexadecimal notation to indicate memory locations and to program instructions to the computer. Writers of the VIP manual assume that you already know how the hex system works, which may not be the case if you are new to computer work, so here are some words of explanation.



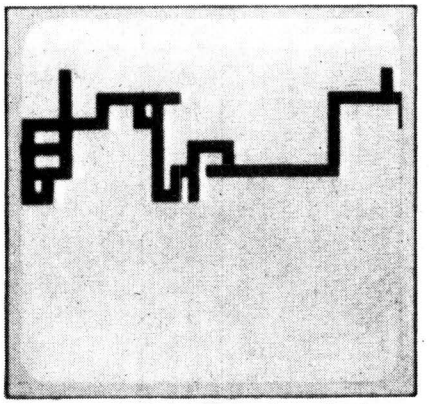
Tick-Tack-Toe—Computerized version of an old favorite—just like the real thing most games end in a draw but the computer will win if you get careless.



Wipe Off—Paddle (bottom center) bounces a projectile across the field and the object is to wipe off the spots in 20 shots.

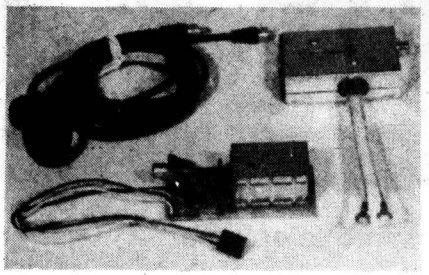


Dot Dash—Keep a moving dot from colliding with the obstructions by adjusting its heading with the hex keys.



A-Mazing—Move a spot through a randomly drawn maze by working the hex keys. The maze can be created by the computer or drawn utilizing the VIP display Drawing Game.

Cosmac VIP



The M&R modulator, at $25.95, is expensive but is factory assembled and includes a broadcast/computer switch and cables.

The hexademical system utilizes a counting base of 16 instead of the 10-base with which we are much more familiar. To expand the decimal counting system beyond 9, without getting into double digits, it is necessary to create additional "numerals" with letters of the alphabet: 0,1,2,3,4,5,6,7,8,9,-A,B,C,D,E,F. To understand how these extra numerals affect consecutive counting, consider memory locations that begin, for example, at 0700. Counting proceeds in familiar manner to 0701, 0702 and so on up to 0709. The next number is not 0710, but instead 070 A followed by 070B, 070C, 070D, 070E and 070F. Only after the full sixteen digits are used, does counting go to 0710 to proceed, again in familiar manner, to 0719 after which you get to 071A, 071B, etc.

Memory locations, as tabulated in various VIP games programs, increase in increments of two. For example:

Location	*Instruction Byte*
0200	6000
0202	6380
0204	611F
0206	620F

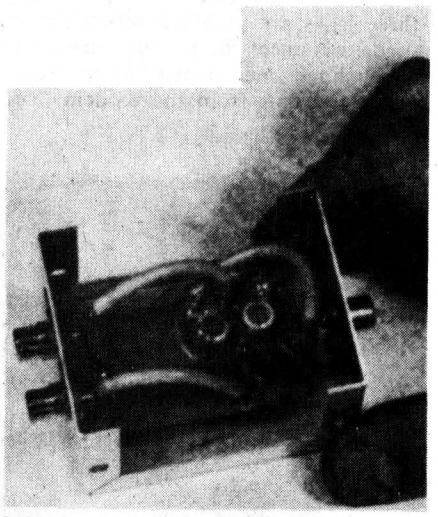
The skipped locations relate to the second pair of instruction digits in each byte. Thus, location 0202 refers to instruction 63 and location 0203 refers to instruction 80.

During normal programming you need only tell the computer the *first* byte location in a sequence to be fed into the memory. Programming a game is basically a simple procedure after the insertion of a CHIP-8 instruction program (more on this later) into memory. Flip the toggle switch to RUN while pressing key C. Punch the four-digit starting location (0200 for games), followed by 0 which tells the computer to start remembering what follows. Key in the whole series of four-digit codes provided in tabular form in the manual. That done, flip the toggle switch to RESET, then back to RUN. If you made no programming errors, the game is ready for play. Should it not work, you can quickly examine all memory locations by a simple procedure explained in the manual.

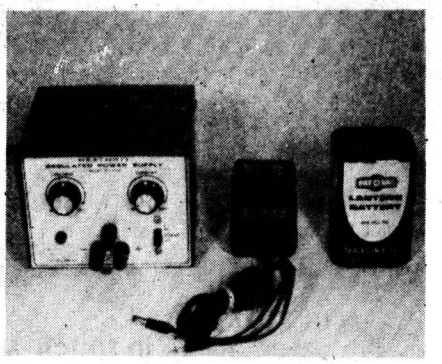
Cassette Recorder a Must. When you turn your VIP computer off, all the information programmed into memory is lost and would have to be reprogrammed. To avoid this tedium, store programmed material on tape from which it can be read back into memory in a few seconds any time it is needed. When recording a new program on tape, I suggest you do so several times in sequence leaving 10 to 20-second intervals between the recordings. If there should be a defect in the tape that spoils one recording, you will still have others that work properly.

Most games require, in addition to the specific games program, a basic CHIP-8 operating instruction program. When you first put this into memory, tape it about ten times on a cassette so that you can keep putting it into the computer memory repeatedly without bothering to rewind repeatedly. Each time the CHIP-8 program and a game program have been combined, and work properly, tape *both* programs in sequence on a tape so that you can at any time read back the *full* program into memory.

When taping a program, or reading it back, you initiate the actions by keying letters F and B respectively, followed by a number indicating the number of 256-byte "pages" to be recorded. It's important to always indicate the length of the program in pages. The last memory byte recorded on tape is shown on the display on completion of a memory insertion from tape; be sure that it agrees with the byte indicated in the manual. Allow two pages for the CHIP-8 program, and one page for every 256 bytes *or fraction thereof* in a game program. If the game has 258 bytes, for example, key four pages.



Of the modulators tested by the author, the Pixe-Verter was the least expensive and worked the best. The assembled modulator should be housed in an aluminum mini-box fitted with phone jacks for the video input and output, and for the —6-VDC power input. The mini-box and jacks are not included in the Pixe-Verter kit.

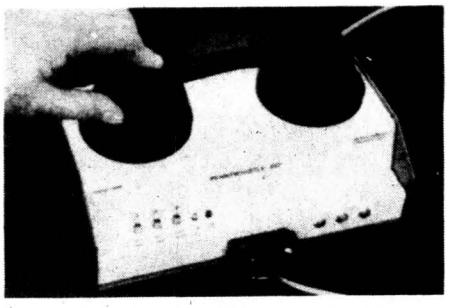


Low voltage power, of either —6 VDC or +12 VDC, is needed depending on the type of modulator used. Six volts can be provided by either a lantern battery, a Universal Adaptor or a regulated power supply. The voltage output should be variable; so add a suitable pot to the lantern battery or Universal Adapter if either is used.

Figuring the proper number of pages can be tricky at times, as in the case of the *Dot-Dash* travel-the-maze game. You must provide enough pages to include memory locations 0300 to 03FF even though they are not used in the basic game (instructions are put into 0200-02FF and the maze pattern is in 0400-0501). So when you record or play back the Dot-Dash game, be sure to key in *six* pages rather than the more usual three to five pages. You can create your own additional mazes by programming into memory block 0300-03FF using another drawing game; but when this information is transferred to the Dot-Dash game, it must be inserted into the 0400-0501 memory bank. These are the kinds of things you must discover for yourself through experimentation because the manual writers do not lead you by the hand all the way.

When you eventually tire of these games, you can go on to a list of more than a score of additional program ideas including such teasers as Intoxication Tester, Shuffleboard, Invisible Maze, Lunar Landing, Blackjack, Turing Machine, Nim and others. The only catch: the VIP manual doesn't tell how to program these and you are strictly on your own. But if you solve even half of these problems, you are sure to become a VIP (Very Important Person) in your local hobby computer circles. ■

HCH assembles the...



Pennywhistle 103 Originate Modem

Keep in touch with a time-share computer via your home telephone.

AS YOU GET MORE INVOLVED in personal computing, meet other computer hobbyists, and join local computer clubs, it's more than likely you'll find there's at least one time-share computer system you can use free, or at a very low cost. It might be the local school's system which is available to students, or perhaps a friend will let you in on his I.D. (identification code), or it might be an older time-share system now underutilized which permits or even encourages outside users for a nominal charge representing the cost of upkeep or repair.

The advantage of a time-share system, in comparison to the average personal computer, is that the time-share generally will have several languages in addition to BASIC: usually Fortran IV and some degree of COBOL. It will also have a file system, and a lot more storage than the average computer hobbyist can afford to build into a home system.

As a general rule, once you locate a time-share you can get on, all you'll need is an acoustic *modem* and your present terminal. You connect your terminal to the modem, dial the telephone number of the time-share system on your regular phone, place the telephone handset in the modem, and you're on-line to the time-share system.

Though most modems are relatively expensive, you can go the kit route and come up with a full-feature model one half to one quarter the price of a commercial modem.

The way to go is with a Pennywhistle 103 Originate Modem. (*Originate* means it's used at the terminal. An *answer* modern is used at the computer.)

In addition to serving as a standard modem, a means of coupling a terminal to a voice-grade telephone circuit, the Pennywhistle has a few extra features that make it particularly attractive for the hobbyist.

Connection is made through a 25 pin D-connector, the type often listed by the "surplus" dealers as RS-232 connectors. Depending on the user selected terminals the connections can be arranged for a TTY (teletype) current loop or RS-232 electrical signal. The TTY loop is through a non-polarized optoisolater so even a hobbyist with no knowledge of how a TTY works, or its connections, can connect to the modem with no hassle. Even the TTY current source is provided by the Pennywhistle, so you don't have to dig through the guts of your TTY to find the current source.

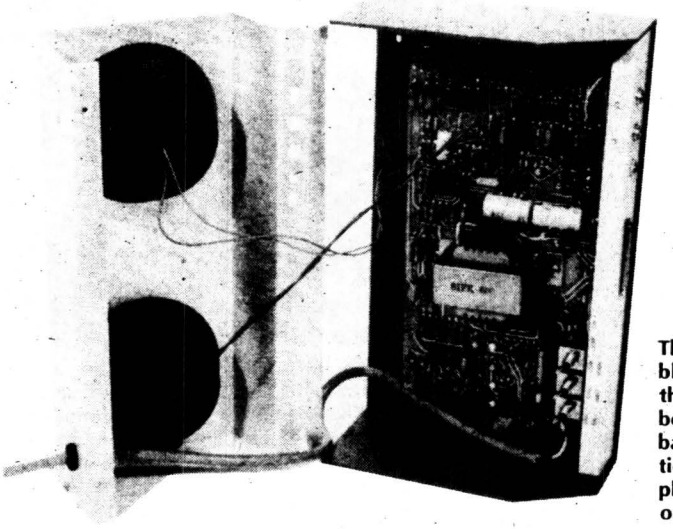
If you have a CRT terminal you will use the RS-232 output. Since the Pennywhistle can handle a Baud rate of 300 (30 characters per second) you might as well use this rate in preference to a lower Baud rate. (There are some hobbyists who run a CRT terminal with RS-232 output at a TTY Baud rate of 110—only Heaven knows why.)

A set of LEDs flicker back and forth when data flows through the modem.

Now for the extras. To start with, three jacks are provided which can be used to connect the modem directly to a telephone line coupler, or to record or play signals from the modem. You can record data produced by the local terminal and data received through the line; you may also play back to the local terminal and to the line.

Another extra is the low/high band switch. Normally, an originate modem transmits on the low band (1070 to 1270 Hz) and receives on the high band (2025 to 2225 Hz). The answer modem at the computer works on the reverse, transmitting on the high band and receiving on the low band. Long distance telephone circuits now have what is known as "echo suppressors" which stop your voice from coming back to you. If you're connected to a time-share through a telephone system with an echo suppressor your terminal's printer is not going to get a signal back from the computer when the terminal is set for *full duplex* (meaning, the printer gets the echo from the computer.)

But with the Pennywhistle 103 you can turn off the echo suppressor by simply flipping the low/high switch to *high* for a few seconds before transmitting data. The modem's output switches to the high band and transmits a signal in the 2025-2225 Hz



The completed assembly just before closing the cabinet. The PC board mounts in the base. The muffs, acoustically insulated telephone holders, mount on the top section.

range. The telephone circuit is fooled into believing it "hears" an answer modem from a computer and turns off the echo suppressor. You then flip the switch back to *low* and use the modem in a normal manner, getting full duplex operation.

The low/high switch can also be used as a *send* switch enabling two terminals to "talk" to each other, though it's to be doubted whether the function would be used by the average computer hobbyist.

Building The Kit. This is not a "beginner's" kit, simply because assembly instructions are sparse. You get a parts list, a *stuffing pictorial* (showing where parts are located on the PC board), and a minimal set of assembly instructions. If you need one-step-at-a-time instructions with a pictorial for every unusual step this is not the kit for you.

One outstanding feature of assembly, which would be welcome in all other kits from all manufacturers, is a direct numbering relationship as to component placement on the PC board. Resistor R1 comes first, then R2, then R3, then R4, etc. (If you're looking where to put R23 it will be between R22 and R24. Same bit with the other components such as diodes and capacitors.) In other words, component identification is in the correct order for the PC board, not the schematic. It actually cuts about 75% off PC board assembly time, and makes it a snap to locate a value swap, such as a 10K resistor for a 100K type.

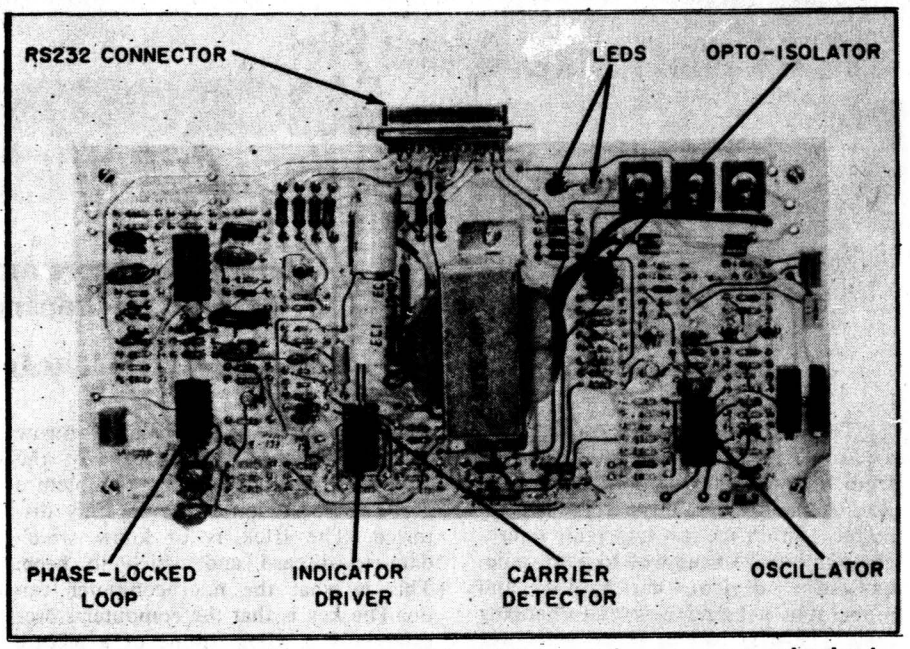
Aligning Your Pennywhistle. Alignment requires a frequency counter and a sine wave signal generator capable of performance to at least 300 Hz. There is no way to get around use of these instruments; without them the Pennywhistle 103 cannot work. Most of the alignment consists of a few control adjustments to get the correct counter reading at a few test points. The procedure doesn't take more than five or ten minutes. The only other main adjustment is to set the output level to the telephone line.

Performance. The Pennywhistle 103 delivers the same performance as our commercial modems—it works. We did find the sensitivity was somewhat high and tended to respond to noises within the room. An engineering note supplied in the instruction manual shows how to lower sensitivity of the carrier detector by changing the value of two resistors. It was an easy enough modification even after the unit was completed, and we suggest it be made if room noises produce false turn-on of the carrier.

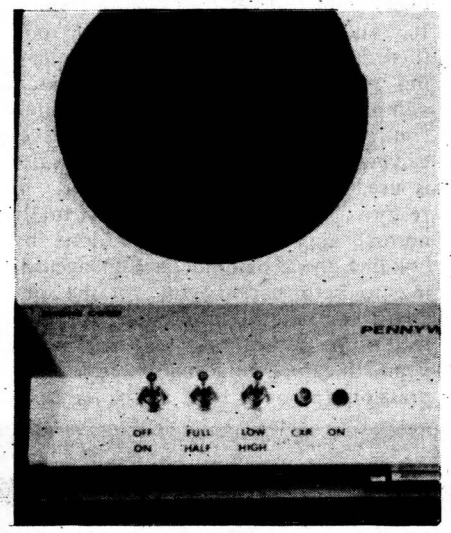
We were able to make recordings of the data signals, and while we have no need for them, nor can see any reason for making recordings of the data signals, nevertheless, the system is there for your use if you have some particular need for the data recordings.

Summing Up. The Pennywhistle 103 works as it should, it's a lot of fun to build (for an experienced builder) and is the least expensive way to modify your personal computer equipment for use on a time-share system. If you have access to a time-share it's time you went for a Pennywhistle.

The Pennywhistle 103 sells for $129.95 in kit form from M&R Enterprises, Box 61011, Sunnyvale, CA 94088.



Virtually the entire assembly is on one PC board. Because of a component order that is in sequence for the PC board rather than the schematic this turned out to be the easiest board to assemble we have ever assembled, and the simplest to debug (swapped resistor values were quickly found). Note the input/output D-connector is part of the top edge of the PC board, saving much trouble.



To the right of the three control switches are two LED lamps labeled CXR and ON. These are the data indicators, which flick back and forth when data passes through the modem. If one locks On then there is no data at all passing through the modem.

Using the modem is simple. You simply dial up the computer, wait until you hear a tone in the handset recever and then force the handset into the muffs. Make certain the handset is firmly seated in the muffs to keep out any extraneous room noises.

MICROPROCESSORS AT THE MIKE

Microprocessors are causing a revolution in the radio transmission and reception

by Ted Koller

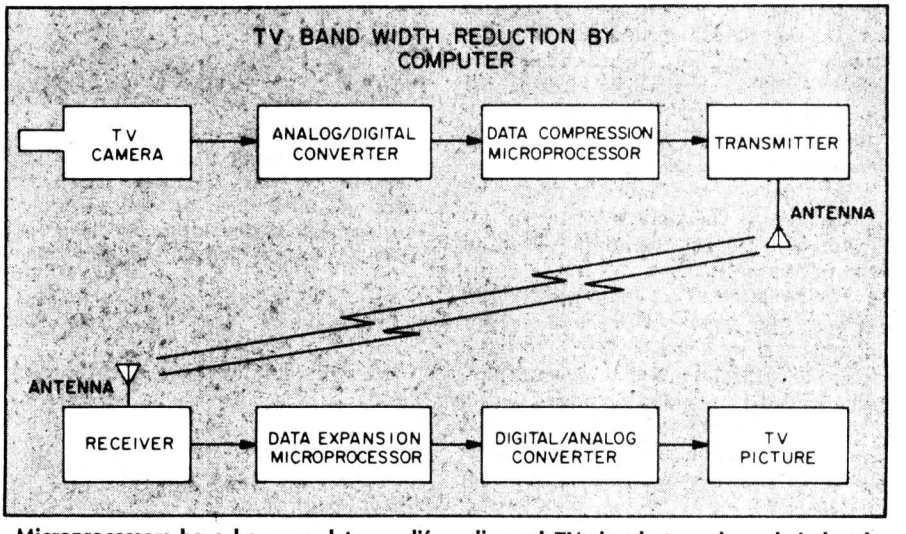
☐ THERE IS PROBABLY NOT a single electronic device that cannot benefit from the addition of a microcomputer. Radio systems of all types are an example. Within the last two years microcomputers have been wed to ham radio, navigation devices, car radios, and more. It is a brand new and changing area. The next two or three years will show not only a whole host of new radio/microcomputer marriages, but also an awareness among the general public that this is just the beginning. In this issue we will explore some of the radio applications of microcomputers and we will look a bit into the future.

Transmission of Radio/TV. Microprocessors have been successfully used to aid in the transmission of radio and television signals. One example is a system developed for the Air Force which uses microcomputers to compress a television signal sent from a camera on a flying missile. The objective is twofold; one, to send a signal which requires less band width than an ordinary TV signal of 4 or 5 megahertz; and second, to make the signal jamproof. Until microcomputers came along, the amount of hardware that had to be mounted on board the missile was a bit impractical to handle. Analog systems were unstable and the digital mini computers really too big for the job besides using a lot of power. Microcomputers available right now offer the needed speed of moving information around, the low power requirements, and the small size.

Here is how the microcomputer compression system works. A TV camera on the missile is to send pictures to a base station where a navigator can steer the missile according to the picture on the base station screen. The TV signal is digitized—that is, sampled at a fixed rate and broken up into voltage amplitude levels that are digitally encoded—by an analog-to-digital converter on board the missile. That signal is then in a form the microcomputer can use because it is really just a stream of 1s and 0s. The computer is programmed to compress that data stream, which basically means it must throw away as much as 80 percent of the data without causing the TV picture at the base station to be severely distorted. The trick is to know which data to discard and which to keep. This is what the microcomputer can do. The key is that the computer stores essentially a whole frame of a picture in its memory, where a frame is made up of 256 lines of 256 picture elements, or data. Then the computer looks for repetition, or nearly equal signals, between lines and between picture frames. If a new signal is just like the old, there is no need to send it. By eliminating repetitious signals, the computer compresses the data and less bandwidth is needed to send it. At the receiving end, a similar microcomputer is used to expand the signal so as to restore the TV picture to a virtually normal appearance. This is done by building the repetition back-in according to information sent by the onboard computer. Jam-proofing, or protecting the transmittal signal from being jammed, can be achieved with this compression approach both in the way the computer can specially encode the new signal before it is transmitted, and in the way the new signal takes up less radio spectrum.

Ham Radio. One of the first microcomputer controlled ham radio transceivers will be marketed by Palomar Electronics (665 Opper St., P.O. Box 2403, Escondido, CA 92025). The predecessor to this unit is the PTR-130K Programmable Transceiver of course also made by Palomar and selling for around $695. That version uses a calculator chip rather than a microprocessor chip, and achieves such features as frequency control and storage of specified frequencies to facilitate automatic frequency selection by the press of a single keyboard button. The new unit, is a good deal more innovative and advanced than its predecessor with a host of convenient features, including a digital LED display—a feature which will soon be standard fare for ham radio operators all over. Before long it will be impossible (I bet) to find an analog frequency dial.

First, the built-in microprocessor controls the frequency you select to



Microprocessors have been used to modify radio and TV signals to reduce their bandwidths and to reduce the effects of interference. Here a microprocessor compresses a signal from a TV camera by rejecting redundant parts of the picture. At the receiver end the reverse procedure occurs under microprocessor control. This sort of system is used in TV guided missiles to reduce bulk and to protect against the jamming of the signal.

Hz steps, or 1000 Hz, etc., and you also specify the starting frequency. The computer will then cause the transceiver to scan up or down from your starting point with the step sizes you specify. If an active frequency is found, it will either pause or camp on that frequency as you wish. The memory unit allows the user to easily recall any frequency at the push of a button and, of course, the transceiver is immediately tuned to that frequency.

Morse Coder. Speaking of ham radios and equipment, several clever hobbyists have, independently, written programs for their microcomputers which automatically translate Morse Code and print the English words on a typewriter. A ham radio converts the carrier frequency to audible dots and dashes that are then sampled by some electronic gates that, in turn, are controlled by the microcomputer. The sampled signals are converted to a digital stream that is then sent to the computer for translation into English. The computer program looks for a relatively long space—which it assumes is a space between words—then goes to its memory to find a match between the sequence of digitized dots and dashes it has received and English letters.

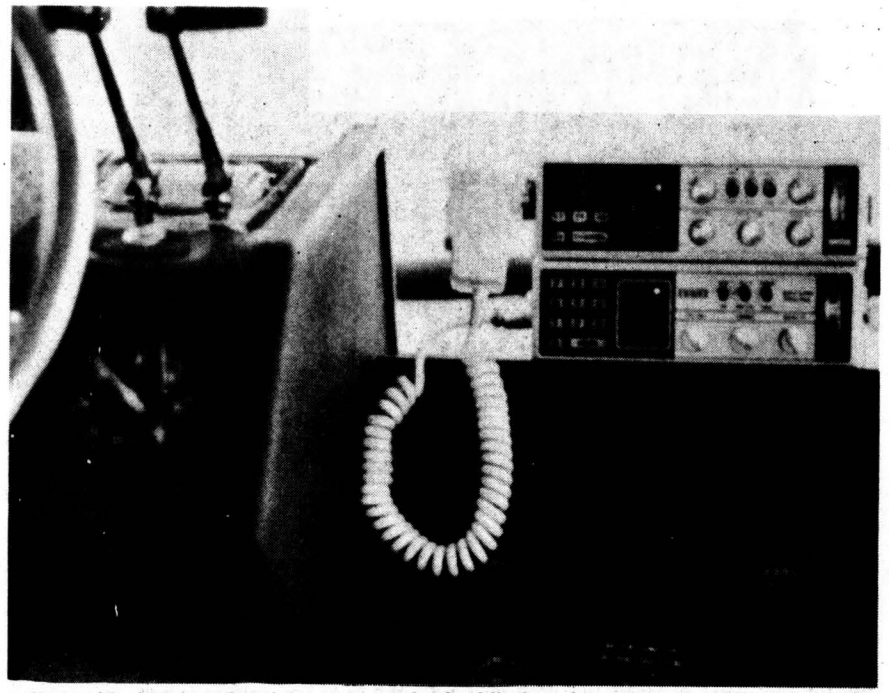
Car Radios. Some 1978 cars, such as Cadillacs, certain Buick models, and Cordobas, have electronic tuning and conveniences, and more cars will have such features in 1979. Chrysler, in particular, uses a microprocessor chip and a keyboard to give the driver a host of conveniences and improvements over mechanically tuned radios. The unit allows you to enter the frequency of any AM or FM station via a keypad. Only FCC authorized frequencies are recognized by the processor chip—so on the AM band, 540KHz, 550 KHz, etc., are allowed, while on the FM, 88.1 MHz, 88.3 MHz, etc., are permitted. The tuner can be instructed to scan up or down the AM or FM bands until a station is found. It will either pause on a station or clamp on it, as you wish. The microprocessor—especially developed for this use—includes a read-only memory (ROM) directly on the chip. That ROM contains the allowable frequencies and the program instructions for scanning up or down.

Navigational Radio. Navigation always requires a computer, whether it is the primitive sextant Columbus used, or just a pencil, ruler and map, or a minicomputer, or a microcomputer. The object in navigation is to find out where you are (at sea, in the air, etc.), what direction you are going, and how long it will take you to reach some point. These calculations require some form of computer. One interesting NASA-sponsored development involved the use of a microcomputer for navigation based on a rather unheard-of radio system called Omega. The Omega system uses very low frequencies (VLF) around 10KHz from eight transmitters around the world. Each transmitter has a specifically assigned set of three frequencies and emits a burst of energy on one frequency at a specified time within a worldwide 10-second cycle before going on to the energy burst at another frequency. Timing accuracy among stations is achieved via atomic clocks. Now suppose you are somewhere on this earth but you have no idea where—however, you happen to have an Omega receiver. It turns out that by receiving signals from any two Omega transmitters and noting the time difference between signals at different frequencies, one can determine that he is on a certain imaginary line, or path, drawn on the surface of the earth. By receiving one more transmitter's signal you can establish another line and where the two intersect is where you are sitting on the globe. As you might guess, this cannot be done unless you have the maps and tables supplied by the U.S. Coast Guard, and the need for storage and translation of such information is where the microcomputer comes in.

By feeding the computer digital samples of the Omega radio signals, it can accurately measure time differences between energy bursts, accurately determine what frequency was transmitted, accurately identify each station being received, make corrections for timing differences caused by humidity and temperature, and—finally—specify where the receiver is located on the globe.

Marine Radio. The Key/Com Fifty Five made by SBE (220 Airport Boulevard, Watsonville, Calif. 95076) was the first microprocessor controlled, keyboard-entry marine radio available, says Gordon West of SBE. Unlike CB, marine radio uses 78 channels in the VHF (156 to 162 MHz) range and only uses FM, whereas CB uses AM and single-side-band (SSB). The Key/Com Fifty Five uses a special microprocessor chip called the Nitron developed for SBE by McDonald-Douglas Corporation. Built into the chip is a list giving the weather channel, and, in addition to the 78 marine channels, 30 police and fire channels. The microprocessor lets the user access these 30 public safety channels for receive only. Further, the Nitron processor unplugs easily, so if the International Telecommunciations Union changes the number of channels, or introduces another channel for emergency broadcasts—or anything else—that processor can be

(Continued on page 82)



Many a ship has foundered or gone aground while her skipper was twiddling the radio dials. These microprocessor controlled marine radios are designed for easy operation. The top radio is a SBE Key/Com Marine CB and the other a SBE Key/Com Marine VHF.

HOBBY COMPUTER=PERSONAL COMPUTER? Though the terms are often used interchangeably, a *personal computer* is generally not the same thing as a *hobby computer*. Recently, *personal computer* has come to mean a complete microcomputer package often supplied with specific software (programing) for small business and educational use. On the other hand, *hobby computing* means the excitement of starting from basics, often building a personalized computer system component by component, much in the same manner that a stereophile might assemble a high fidelity system.

As with most action-type hobbies, there are many dimensions to hobby computing, with costs ranging from rock-bottom to astronomical. At the bottom are basic computer kits that serve as instructional devices, or introductions to computer technology. At the top are full-blown computer systems that rival the commercial minicomputers in common business use. There is hobby computer equipment suitable for a grade-school child, and equipment suitable for the graduate computer-science student. Much hobby hardware can often be put to work bringing in extra sideline income, while others utilize hobby equipment for straight commercial use. In short, there's something for just about everyone in hobby computing.

The Heart of the Computer. Whether we talk of a basic "beginner's" kit, or a full-blown computer system complete with mass data storage devices and line printers—called the CPU for Central Processing Unit—the heart of the unit is often the same relatively inexpensive integrated circuit.

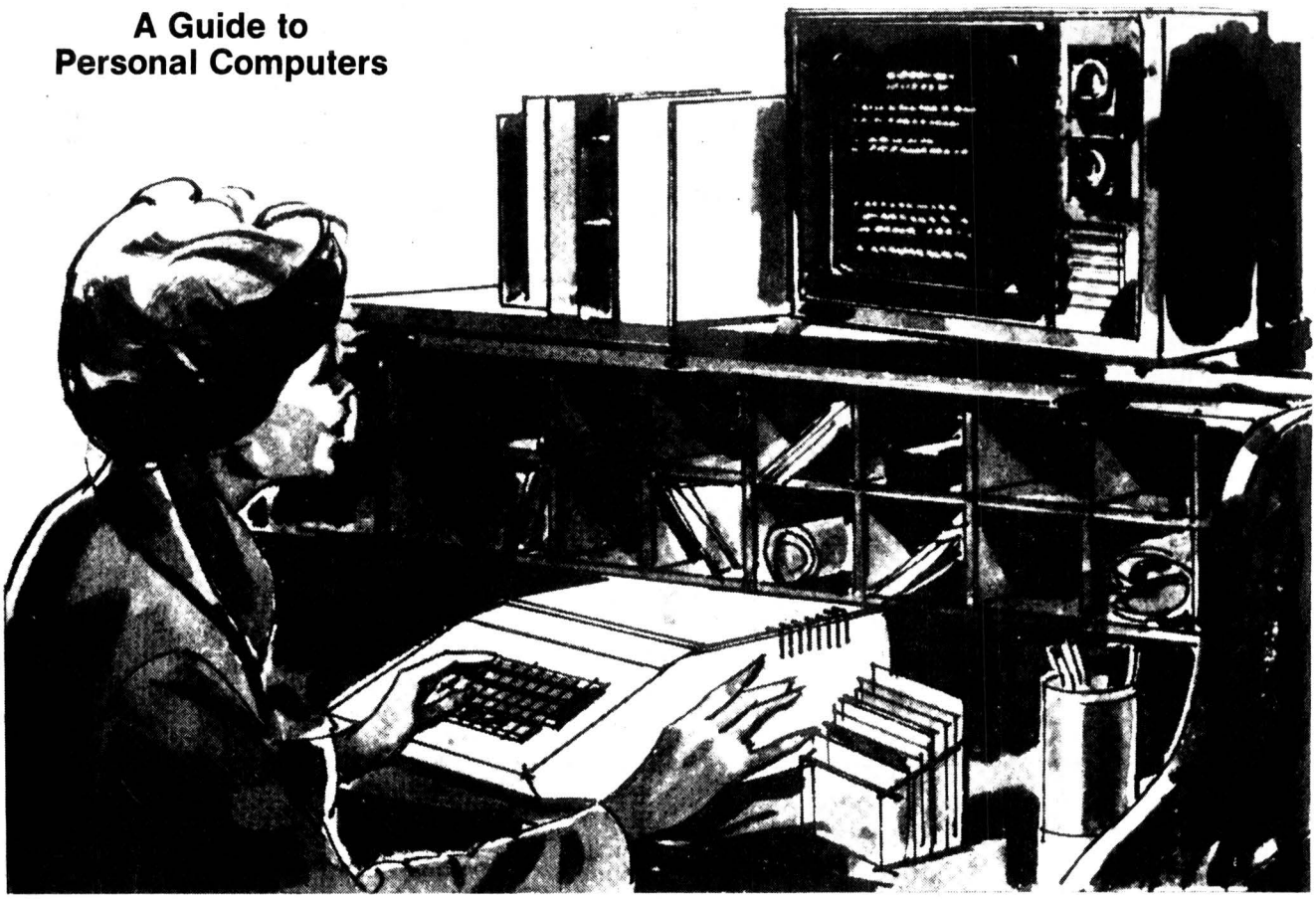
Several manufacturers incorporate within the CPU with some form of readout and a minimum of "memory" in what we term a "trainer," a low cost computer kit primarily intended for getting a solid foundation in the basic principles of how a computer works. Except for the RCA-based "ELF" trainer which can be expanded with additional memory and a recorder *interface* that permits data and program storage via an inexpensive audio cassette recorder, the trainers rarely can be used for anything more than as a low cost introduction to computers.

But take the same CPU, add at least 8K of memory (actually 8192 bits but we always refer to memory in units of 1K, such as 4K, 8K, 16K, etc.), provide for connection of either a TV or printer terminal, program the computer with a high level language such as BASIC, and we have a full-blown computer that can be used for everything from school math, to business applications, or "arcade games."

For example, both SWTP Co. (Southwest Technical Products Co.) and Heathkit have build-it-yourself basic computers which consist essentially of the CPU, a regulated power supply, a minimum amount of memory, a built in (resident) operating system, and a provision for plug-in memory expansion

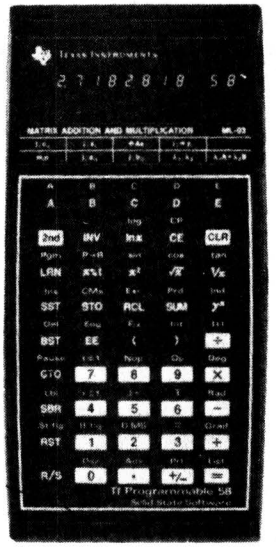
Hobby Computer Handbook

A Guide to Personal Computers



HCH checks out the...

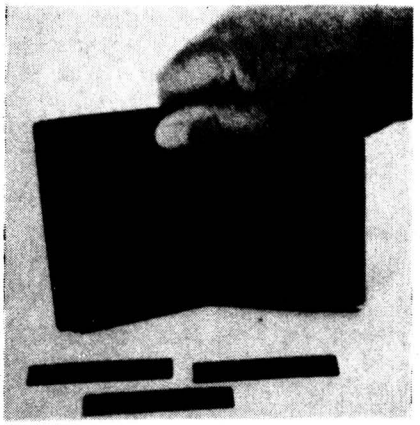
TEXAS INSTRUMENTS 58 PROGRAMMABLE CALCULATOR



It's almost like carrying a computer in your pocket, when you handle one of the new generation of electronic calculators.

IMAGINE A COMPUTER SO SMALL it can fit the back pocket of your jeans, or the inside pocket of a jacket. Not possible? Wrong! Though Texas Instruments, Inc.—or TI as they are more commonly known—has decided to call their model TI-58 a *programmable calculator* it is in fact a *computer*, with a lot more capability than some of the devices presently sold as "full scale computers."

The TI-58 resembles many other scientific type calculators such as TI's own models 50, 51, 56, etc. It has a twelve character LED display that shows 10-digits with sign in standard format, or 8-digit mantissa with 2-digit exponentiation and 2-signs in scientific notation. The keyboard has the usual



A complete set of label cards for each module fits into a compartmented wallet (supplied) which, in turn, fits inside a compartment in the calculator's soft-case.

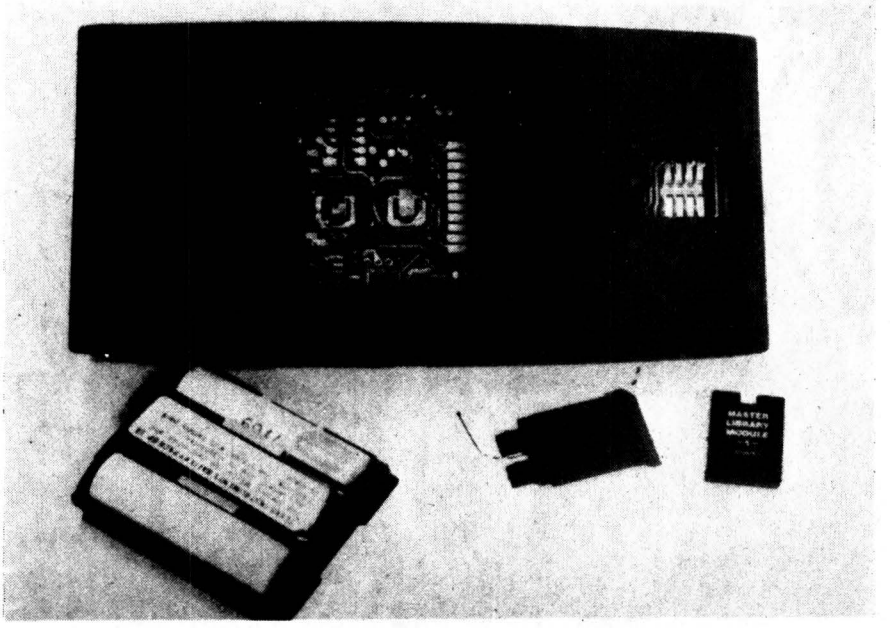
assortment of transcendental functions, storage (memory), parenthesis, etc., etc. The calculator uses an Algebraic Operating System (termed *AOS*) which means you enter the problem just as its written—usually from left to right—as you did (or do) in school. You don't have to learn a new way to express or work mathematical relationships.

In addition to common scientific features the TI-58 can be programmed just like a computer for up to 480 steps. Actually, the number of steps is determined by the amount of memories needed as the overall capacity is shared between program steps and memories. If you need complete memory, and no steps, up to 60 memories are available. User partitioning allows the memory and program steps to be split as needed. You can partition the TI-58 for 320 steps and 20 memories, or 480 steps and no memory, and every choice in between.

It's true there are other programmable calculators that approach the TI-58's steps and features, so you're entitled to ask "Why call it a *pocket computer?*".

The Big Plus. Well, it's the extra features in the TI-58 that make it a pocket computer. Among these extras are: ful editing capability; data memories; test registers available for looping, increment and decrement; and most important, solid state software (programs) available on thumbnail size ROMs (read only memories) that plug directly into the TI-58, and which can be utilized as independent programs or as part of a larger user-written program.

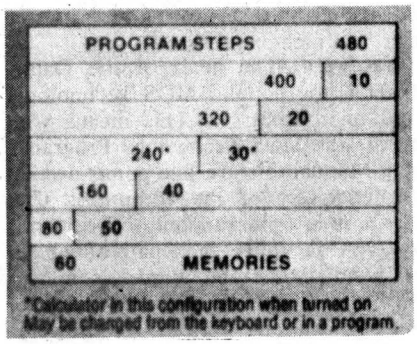
Using these extra features is ably taught in the instruction manual, *Per-*



The module, which is no larger than the tip of your thumb simply drops into a polarized opening in the rear and is secured by the sliding door. To change modules you lift out the one that's installed and drop in the desired one. You cannot install a module incorrectly. When the battery pack is removed, internal contacts connect the TI-58 to the matching printer/plotter by simply securing the calculator to the printer. Because there are no connecting wires the calculator is restored to non-printer use by simply removing it from the printer and snapping the battery pack into place.

TEXAS INST 58

sonal Programming, which is packaged with each calculator. Much more than just an instructive pamphlet, the book runs to more than two hundred oversize pages and is—in itself—a fine entry into the world of programming. With this manual, coupled with the features of the calculator, the user will find himself able to program for sophisticated, complex problems within a very short time.



With the TI-58, program steps and memory registers can be varied by the user. The 58 has a 240-program step and 30-memory configuration when turned on. This can be varied at the keyboard or in a program.

Firstly, *full editing.* Assume you have writen a program and then discover that in entering the program you have left out a step, or you want to change one or more steps. With the TI-58 you don't have to re-enter the program. Using the editing feature you can make changes, insertions, and even move the program forward, just as you might do with a computer.

The whole business of editing has been reduced to a series of easy, keyboarded steps. Four keys control the entire process. The calculator is first placed in the Learn mode; at this point, the LED display gives a numerical indication of the program step along with a label as to what that step is about.

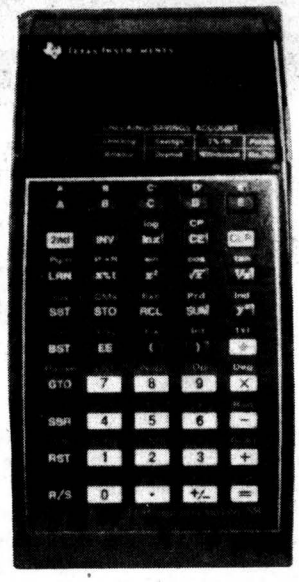
For instance: 000 76 on the LED display reveals that at the beginning of your program (000) you hit the Label (LBL) key. (76 is the position of the LBL key on the keyboard.) Likewise, 017 01 means that as the seventeenth step of your program you have entered the number 1.

You can single-step right through the program, and you can backstep and return to any point. Once you have found where you want to make the change you have only to key it in.

The *test register* actually permits you to test for *less than, greater than, equal to,* and *not equal to,* and then loop or shift to a function or subroutine depending on the test. Again, this is a computer-type function.

One of the most useful keys in making your program is the DSZ key, which stands for 'Decrement and Skip On Zero.' This is the key which allows you full control as to how many circuits of a 'loop' a program should make.

Instead of having to run the same program a few times over by manually pressing the Run/Stop key and keeping track how often its pressed, you just have to sit back and allow the DSZ to do it all for you. The convenience and sheer powerfulness of it all is bound to lead you deeper—and more rewardingly—into programming than you had ever thought you might go.



The Texas Instruments Programmable 58 calculator with an instruction card being inserted. Every subroutine contained in the calculator's Library Module has a separate card. The cards tell the user how to call up and use each individual program, so that there is no need to refer to a book.

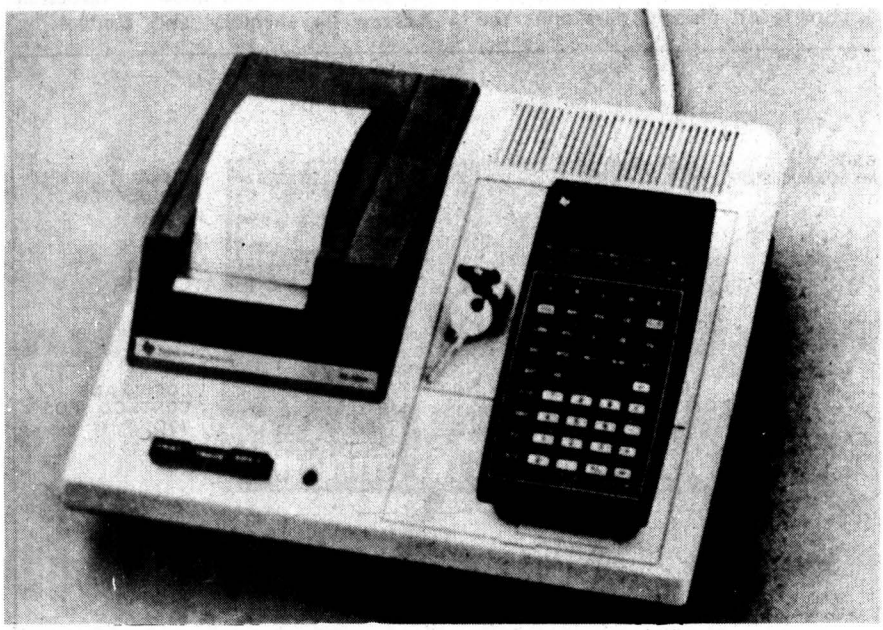
An Elephant's Memory. Every TI 58 has the capability of over 5,000 extra steps stored on a ROM (*R*ead *O*nly *M*emory) chip which plugs into the back of the calculator. TI calls this ROM a module, and they have an entire library of such modules.

The master module, which is supplied with the calculator, consists of twenty-five preprogrammed functions. The listing of programs (or subroutines) is truly astonishing; Matrix Inversion, Zeros of Functions, Polynomial Functions, Simpson's Approximation Continuous or Discrete, Compound Interest, Annuities and Moving Averages to name but a random sampling.

The great thing about this, is that each of the on-ROM subroutines is just that—a *sub*routine. You can call them up as part of your own program, use them in a loop, have them operate within whatever parameters you choose.

If you need even more versatility, TI has five other ROM modules programmed for specialized areas. These optional modules contain subroutines

(Continued on page 81)



The PC-100A printer adds alphanumeric printout to either the TI-58 or the 59. With this capability the distincton between calculator and computer becomes a very fine line. Installing one of TI's calculators on the printer is quite simple. The battery compartment is removed and the calculator then fits right onto the PC-100A.

HCH checks out...

DATAC 1000T COMPUTER

Nuts and bolts computer board instructs as well as computes—economically

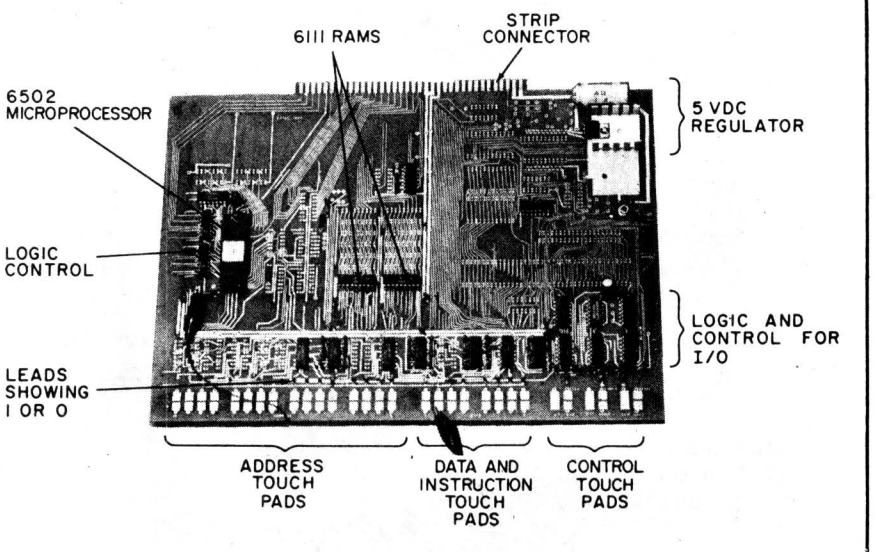
☐ **Two of the most interesting things in the microcomputer field are the new devices that let you talk to your computer, and the very basic microcomputer training devices. In this article we will talk about each of them —the current work in the area of talking to computers, and an interesting training computer from an innovative company called DATAC.**

DATAC. Datac Engineering is a rather new and growing company with an interest in designing and manufacturing microcomputers students and hobbyists can use and use quickly. The Datac 1000T is a tutorial (thus the "T") microcomputer on a 9 X 12 inch printed circuit board. It has no fancy keyboard or numerical readout, but is just a basic computer with the simplest possible input and output arrangements. The input is provided by the user one bit at a time just as a computer would operate internally. To provide these bit-at-a-time inputs, there are small metal touch pads along the bottom edge of the board that are touched one at a time by a probe. Each address word is 8 bits, and each data word to be stored in an address location is 8 bits. For each address bit there are two metal touch pads on the printed circuit board, one pad for "0" and one pad for "1". So by touching the appropriate pads one can write any binary address or data word. If you want to get to address location 0000 0100 (location 8) in memory, just hit the zero pad for the first binary digit, and so forth until all 8 bits are entered into the control unit. The readout that lets you see what digits have been entered into memory, and also shows the data that you are putting in or are getting out, is made up of small LEDs above the touch pads. When you probe the address memory touch pads with 0000 0100 the eight LEDs for the address will all be out but for one and this display indicates 0000 0100. This way you know the address location the microcomputer is using, and for data you know the values stored in memory. With some of the tutorial microcomputers, like KIM-1, you have to read hexadecimal code on a numerical display; here you read hexadecimal code from individual lights. Remember that 0000 is zero, 0001 is one, 1001 is nine, 1010 is called "A", and so forth until 1111 which is "F".

While on the subject of the input and output, let's take a look at some other touch pads and corresponding LEDs on the Datac 1000T. In the lower right-hand corner are nine pads used to give commands to the microprocessor unit. When Reset is touched by the probe the processor finds the beginning of your program and awaits instruction to run through it. The Run pad executes the program while the Norm pad (touched after Run) causes a regular lightning speed computer run, and Single causes the computer to execute one instruction with each touch of the probe to the Step pad. There is an Examine pad for fetching contents out of memory, a Load pad for placing data into the processor, and a Deposit pad for storing data into memory cells.

Hardware. The heart of the Datac 1000T is the popular MOS Technology 6502 microprocessor. This means you can use the MOS Technology Programming Manual to learn fine points on the instruction set of the computer. The storage is in 6111 random access memory (RAM) chips, each pair of which can handle 256 8-bit words. You can see that with an 8-bit address word, 256 memory locations (2 to the 8th. power) can be reached. This block of 256 locations is called a page of memory (like a page in a book).

The Datac 1000T has been designed with expansion in mind. More pages of memory can be added by slipping extra 6111 RAMs into the pre-drilled holes on the printed circuit board. In fact, the microcomputer is so basic it could have been built on a board one-half the size, but the present size allows a threefold increase in memory and control by



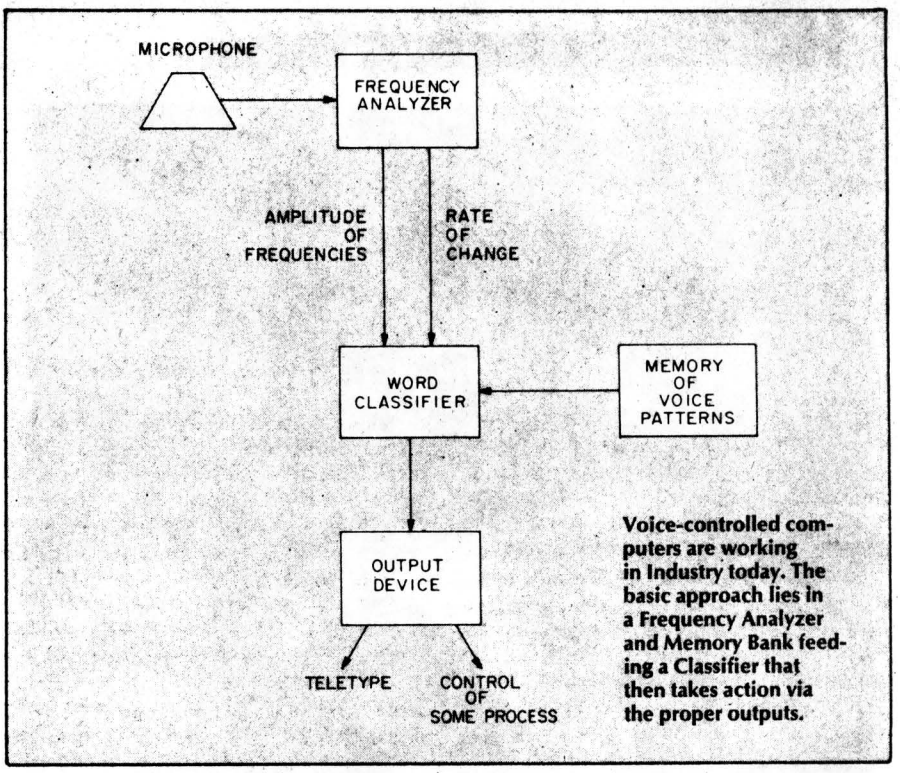
The Datac 1000T does not have a keyboard or a numerical readout, but it's a fine introduction to computers and to programming. Programs are inputed via unique touch pads which hexadecimally address any data word. While more tedious than a keyboard, this method is one of the best to teach the user what is going on inside a computer.

addition of the appropriate integrated circuits and memory chips. Power for the unit is supplied by a small wall plug unit. Finally, there is a connector strip at the back of the PC board for access to control functions in future Datac versions. Readers interested in more details of expansion capabilities and options should write directly to Datac Engineering at P.O. Box 406, Southampton, Pa. 18966.

Make It Work. Let's do a simple programming example with our Datac. We will write and enter into memory a program to add two numbers. Such a program is the beginning of many more complex and very useful programs for controlling railroad trains, playing computer games on TV screens, and the like.

We will begin at location ten in memory (there is only one page in the Datac 1000T so we do not have to specify pages here) by placing instruction "D8" there. This hexadecimal instruction has special meaning to the 6502 microprocessor as we shall see shortly. To write D8 at location ten simply touch the probe to the pads in such a way as to light the LEDs to read 0001 0000, 1101 1000 (10, D8). Now touching the Deposit pad will place instruction D8 at location ten. The program continues as shown in the box accompanying this article.

Examine the program. First, the D8 instruction tells the microprocessor to add numbers in binary form so we can read out the hexadecimal answers on our display. The next command ("18") sets a carry-flag register to zero just so the computer will not think something is left over from a previous addition that it has to add in here. Then the two selected numbers are read-in, added, and placed in memory slot 00 where the final answer will be found. The next two commands actually tie-up the computer doing busy work so that it will not by chance take noise bits from the memory as commands, operate on them and perhaps (really little chance) destroy the result in location 00. These two steps cause the computer to go back and forth between locations 18 and 19 so it can get out of the loop. When you hit Halt, the looping is stopped. That's all there is to it! The last line of code is used to tell the computer the starting point (memory cell ten) of the program when Reset is touched. This microcomputer can handle a wide range of interesting programs, especially games you can invent yourself, because it is based on the powerful instruction set of the 6502 processor. The Datac 1000T unit costs $185 and can be purchased by writing the address given above.



Voice-controlled computers are working in Industry today. The basic approach lies in a Frequency Analyzer and Memory Bank feeding a Classifier that then takes action via the proper outputs.

Programming the Datac 1000T

11	18	Clears the carry bit.
12	A9	Place number below into accumulator
13	12	First number to be added (any number)
14	69	Add number below to accumulator
15	24	Second number to be added (any number)
16	85	Place accumulator value into location 00
17	00	
18	4C	Jump to location 18 of memory
19	18	
7C	10	When Reset is touched, go to location 10

To run the program hit Reset, Run, Norm, Halt, then touch Examine and the address pads for location 00 (hex). The answer 36 (24+12) will appear on the LEDs. (See text for more explanation.)

Number Please. We just talked about a very simple, basic computer; now let's go to the other extreme and learn about talking to computers, and computers talking back. Ten years ago developers began to think that the wild dreams of carrying on conversations with computers might just be possible someday. While dialogue with a machine may be decades away, there are over 150 special purpose electronic computers today that are in operation receiving voice commands. The first installations of these units were in the airline industry during the early 1970's. Voice-controlled machines would operate baggage sorting equipment by steering suitcases to appropriate locations under selected voice commands. With microprocessor power and flexibility increasing at a rapid pace, voice-controlled units can be expected to blossom.

The advantage of a voice-controlled machine lies in the mobility given to the user; it allows hands and eyes to be focused on the task rather than on the computer keyboard. A basic speech-recognition system has a microphone feeding a Frequency Analyzer whose output gives both the amplitude of the various frequencies in the voice command and the rate of change of those amplitudes. Two persons saying the same word will have different frequency content and it is the job of a Word Classifier to recognize the word based on three inputs. It receives the two pieces of frequency content information from the Frequency Analyzer and it receives reference information from

(Continued on page 82)

HCH checks out the...

Find programming gold at the end of this Apple's rainbow.

Many hobbyists just starting out in personal computing learn the hard way, and much to their pocketbook's chagrin, that a "basic computer kit" is just a black box that also needs a terminal, memory expansion, higher language program, and some device to feed the higher language into the computer. Generally, the "extras" cost more than the computer itself. Other hobbyists learn that some kit computers have little or poor technical backup for the builder who runs into assembly problems, or who receives defective components.

One way to avoid all these hassles and get right into the end use of personal computing is to purchase a complete system that's up and running as soon as it's unpacked, preferably with a built-in higher language such as BASIC. One of the personal computers that nicely meets these requirements is the Apple II from Apple Computer Co.

Skin of the Apple. The Apple II contains so much in the way of resident programming and operating features it's difficult to know where to start, so let's begin with appearance.

The Apple II is built into a low profile desk-size cabinet slightly larger than a portable typewriter and it weighs only a bit more than a good quality portable. All circuits are self-contained right down to the keyboard, so the first logical question is: "What is used for the display?"

The Apple II has a video jack for a CRT (video monitor), but the company recommends that you purchase a RF modulator and use your own TV rather than an expensive CRT. Just about any of the popular RF modulators will work well with the system and you'll find that the quality of the picture will rival even the best CRT.

Since the Apple II has built in color graphics the best video monitor is a color TV, though you can use a B & W if you have no need for color graphics or are trying to cut costs. (Right off the bat you start saving because there's no need for a relatively expensive CRT monitor; your home or office TV, no matter how inexpensive, can serve as the monitor.)

The output of the RF Modulator feeds a coupling device such as the ones supplied with electronic games; the type that lets you switch the TV between the game (computer) and the regular antenna. Even with the modulator installed you can still use the video output to drive a standard 75-ohm input CRT monitor—if you have need for such a device.

Another extra supplied with the Apple II is a set of "game paddle controls" which can be plugged in by the user. Apple has some game programs available on cassette tape, or you can write your own. The computer works with or without the paddles, there's no need for modifications or "jumpers"; if you want the paddles you just plug them in.

Moving along to the most important part, the computer itself, we find it is initially supplied in the most standard version (sells at $1,045 with 8K of RAM and 8K of ROM (*R*ead *O*nly *M*emory). The ROM contains a 2K monitor system and a 6K BASIC that includes any length variable names, syntax and range errors indicated immediately when entered, multiple line statements, graphic commands (COLOR = expr, PLOT,HLIN,VLIN, etc.), paddle read commands, TEXT and graphic commands to set display from BASIC, memory boundary adjust, switchable I/O (input/output) assignments, cassette save and load commands, and a host of other usual and unusual features.

However, the resident BASIC is an integer BASIC (no decimal places). While it can be used for many advanced programs it is less than suitable for a student as far as math or statistic studies are concerned. Apple's solution?

If you do need a floating point BASIC, Apple has available a 16K BASIC on cassette tape (which requires 16K of memory) that is among the most complete we have seen or used in personal computers. Perhaps the only limitation—caused by the fact the Apple II is a microcomputer and there's a limit to what can be built in at reasonable cost—is that each time extended BASIC is loaded the user must decide between having graphic commands or LET and REM commands. The extended BASIC also provides for moving the cursor anywhere on the screen without affecting anything already displayed there; or, the cursor can be programmed to erase as it moves, or to enter displayed characters into memory.

We could use up several pages describing the programming features and still might not cover everything of specific interest to you. The best thing is to send for Apple's descriptive brochure which spells it all out in detail.

One important thing we would like to mention is a special appendix in the "APPLESOFT Reference Manual" (for extended BASIC) on converting BASIC programs not written in APPLESOFT.

Apple II Computer

One of the most frustrating things for the personal computing hobbyist is to spend an hour or more typing in a seemingly endless program only to find it doesn't work because there is little standardization between so-called *ANSI Standard BASICs*. Between ANSI BASIC and IBM BASIC, there are some far out discrepancies. Apple's conversion appendix can't handle every problem you might run into, but it does cover a goodly number of the most common discrepancies.

One thing we must mention is the sound effects generator, audible through a built in speaker. The generator can be user-programmed to *pong, ping* or *flutter* with a variety of tones. Really enlivens games!

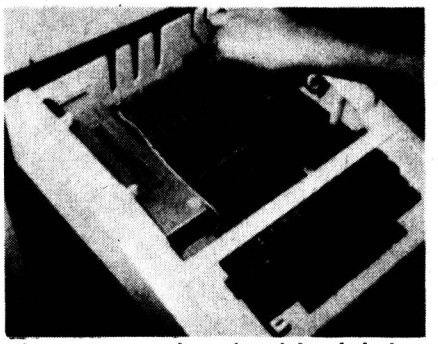
At the Apple's Core. The computer can address from 4K to 48K bytes of RAM on the main board, and sockets are provided so the user can increase memory by simply plugging in the appropriate chips. Either 4K or 16K chips can be used. The total RAM capacity is organized into three increments and you can intermix 4K and 16K chips. Memory might be configured as 4K/4K/4K, 16K/4K/4K, 16K/16K/4K, etc. It can be ordered with any initial configuration, factory installed and checked. Memory select sockets and "jumpers" program the computer for your particular memory organization, and the jumpers can be changed easily if you upgrade the memory capacity.

Just as the user can add RAM by simply plugging in the appropriate ICs, so too can he add ROM by plugging programmed ROMs into one or both of the spare ROM sockets factory installed on the main board. The address for the ROMs (or PROMs) is above the highest RAM location and can be under software control, as can be the built-in monitor and integer BASIC.

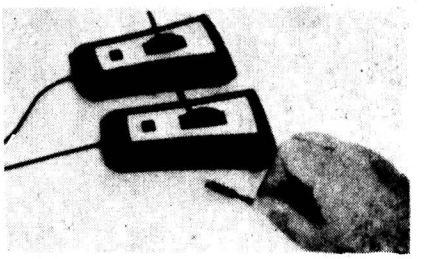
While there are eight connectors for peripheral equipment such as printers and disk recorders, no peripheral control boards were available at the time this article was prepared. Cassette recording of programs through the SAVE and LOAD functions are presently done utilizing an ordinary audio cassette recorder through phono jacks labeled cassette IN and OUT. The OUT feeds a recorder's AUX input. The recorder's monitor output (from across the speaker) feeds the IN jack. The recorder interface operates at 1500 baud, some five times faster than the 300 baud Kansas City standard used by many hobby computers. This permits unusually rapid loading of long programs such as the 16K Applesoft extended BASIC. Note that the recorder is not under computer control as far as start and stop is concerned; the user starts the recorder in the play or record mode before hitting RETURN (to load or save), and stops the recorder after the operation is completed.

We had no problems loading the sample game, graphic and checkbook programs using a moderately priced cassette recorder (about $50), and could load the 16K BASIC tapes (we

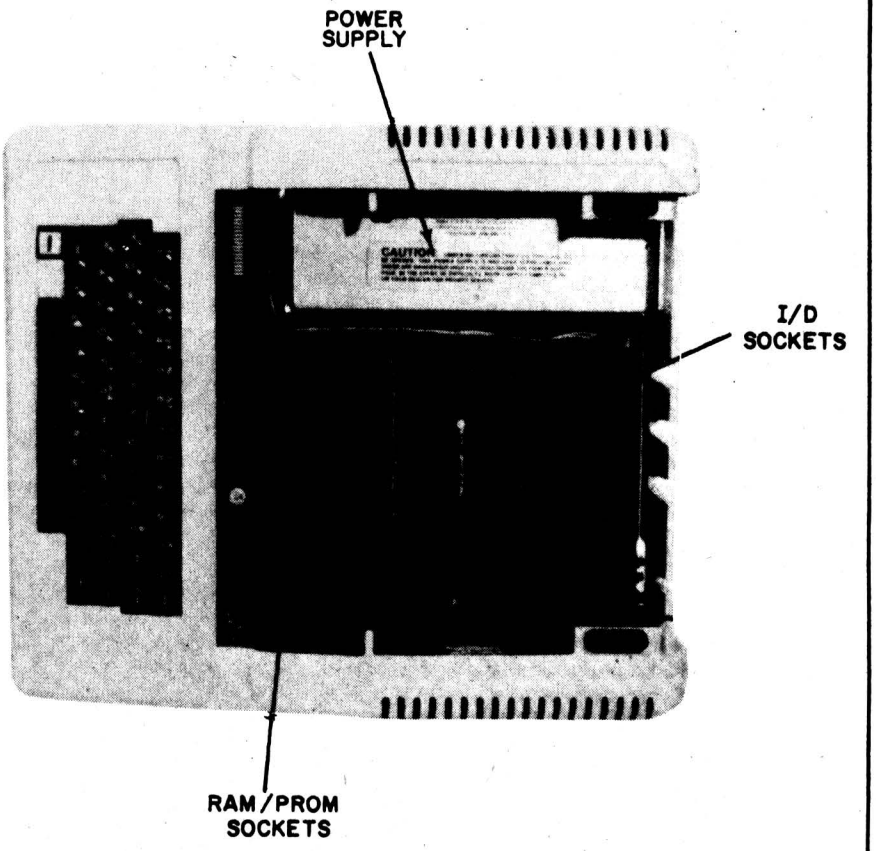
(Continued on page 81)



There are a number of peripheral devices available from Apple. They plug right into I/O ports provided in the back of the computer's board, tucked away in cabinet.



Home computers often are used for games —they can really be a lot of family fun! With many home computers it can be technically difficult to add game paddles, which somewhat limits the games that can be played. Apple comes equipped with two paddles, which can be plugged right in and programmed to operate with special BASIC commands and functions.



The Apple II is a desk-size computer with a built-in keyboard, integer BASIC in ROM and 4-color display capability. It has a standard video output jack for connection to either a color or black-and-white monitor. However, you can save a good bit of money by using a simple RF modulator to connect the Apple to your home television set—and the display quality will still be excellent. You can add more memory, up to 48K, at any time. If you need more than an integer BASIC, Apple has a 16K BASIC available.

BIO LOGIC

The ultimate personal computer is inside your head.

by Walter Sikonowiz

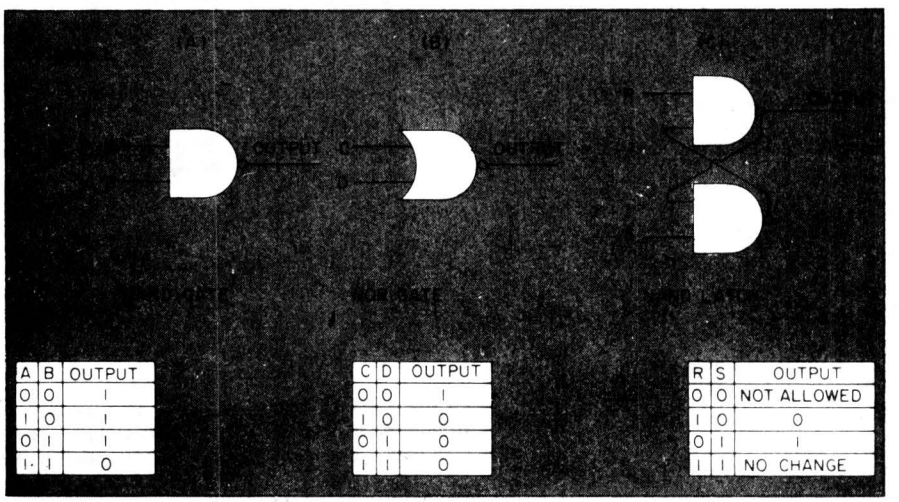
It seems that the day of the home computer has at last arrived. Everywhere, we read about these new machines that are changing our lives, running things better than we ever could ourselves. Everybody tells us that the solid state computer is the newest and most radically advanced thing to come along since the sun. But, you know, in spite of all these claims, the personal computer is not all that new. In fact, you've been using one all your life; it's called a brain.

Many people, electronics hobbyists in particular, have long suspected some fundamental similarities between brains and electronic computers. In this article we're going to compare these two types of information-processing systems.

Let's begin on familiar ground. In Figure 1 we have the various logic gates; the important building blocks of any electronic computer, big or small. The NAND gate's output state is a function of its two inputs, and it remains at a logical 1 so long as the input state is not 11. An input of 11 sends the output to a logical 0. On the other hand, the NOR gate's output is usually a logical 0 except when the input is 00, in which case the output assumes a logic 1 value. Using combinations of just these two gates we can synthesize any desired logic function, no matter how complex.

As you know, the 0 and 1 symbols can denote voltages, with 1 usually representing a high voltage and 0 a low voltage. The ones and zeros may also be interpreted in a philosophical sense as True and False, respectively. Why should we use numbers to represent qualities such as truth and falsity? It's mainly a convenience which allows logic to be expressed in mathematical form. If logic can be handled algebraically, then solution of logical problems becomes routine, and can be handled by machines—computers.

But a computer needs more than



A	B	OUTPUT
0	0	1
1	0	1
0	1	1
1	1	0

C	D	OUTPUT
0	0	1
1	0	0
0	1	0
1	1	0

R	S	OUTPUT
0	0	NOT ALLOWED
1	0	0
0	1	1
1	1	NO CHANGE

Some Basic logic circuits: a) A NAND gate gives a 1 output for any input except 11. b) the NOR gate only gives a 1 output when both inputs are 0. The NAND latch (c) is a memory device which is the basis of the common flip-flop circuit.

decision-making apparatus (NAND and NOR gates); it needs memory. We store the computer's instructions (program) in memory, and also use memory to retain the solutions generated by the computing process. One basic form of memory is the NAND latch (Figure 1c). When a little control gating is added to this basic circuit, we can get various flip-flops, which are probably the most familiar storage elements. Other forms of storage include magnetic cores, magnetic tape, paper tape, and the exotic magnetic bubble devices. Regardless of the storage form, however, memory simply holds data until t's called up for processing or readout.

Neurons. Contrast the human information-processing system to the electronic computer. Like a computer, the human brain is a complex entity built up from enormous numbers of simpler fundamental units. These basic units are called nerve cells or neurons. Several classes of neurons exist; neverthless, all share the same generic traits. Figure 2 shows one typical neuron. The tentacle-like elements are called dendrites, and they function as the input leads to the cell. The single long filament leaving the cell is an axon, which is the cell's output. Nerve signals travel along the thin membrane that encloses the cell's protoplasm. Adjacent cells communicate with one another across synapses—gaps between axons and dendrites of different cells.

Logically the neuron is more complex than a NAND or NOR gate. It functions as a majority gate, which may be described as follows: Suppose, for example, that we construct a gate with five inputs and one output, and that the output will be a logical 1 whenever a majority (3, 4, or 5) of the inputs is in the 1 state. If there are less than a majority (0, 1, or 2 that is) of logical 1 inputs, the output is a logical 0. Such majority gates are not encountered too frequently in electronics, one notable exception, however, being Motorola's CMOS MC14530 dual 5-input majority gate. As a rule a neuron has more than five inputs—typically, up into the hundreds. Regardless of its complexity, however, a majority gate can be expressed in equivalent form as a combination of NAND and NOR gates.

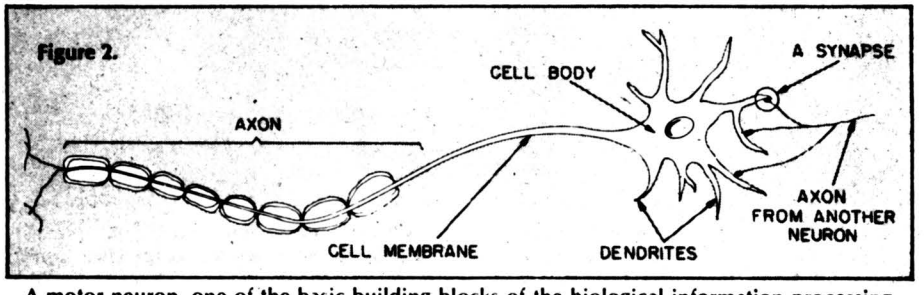
From the foregoing you can see that computers can be logically equated to neuron networks; nevertheless, certain physical differences exist between biological and electronic systems. To begin with, electronic signals are viewed as differences in potential, which govern the flow of free electrons. Signals in nerve nets, however, consist of waves of polarity reversal on the surface of polarized neuronal membranes. In Figure 3 you can see that a nerve's membrane normally possesses excess positive charge on its outer surface, and excess negative charge on the inner surface. Excitation of the nerve brings about a polarity reversal at some point on the membrane, which then induces polarity reversal at adjacent points on the membrane, and the signal spreads. The reversed-polarity condition at any given point reverts to normal after several miliseconds, while the wave of polarity reversal propagates to new points on the membrane. Charge distribution on the membrane's surface is controlled by movements of sodium and potassium ions, and the rate of ion flow through the membrane limits the speed at which nerve signals travel to about 100 meters/second. Contrast this with the speed of an electronic signal traveling down a transmission line with a polyethylene dielectric: about 2×10^8 meters second. Biological systems are thus inherently slower than electronic ones.

Memory. Two physical characteristics distinguish the human nervous system from the electronic computer. Communication between the functional units in a computer is established by connecting wires. Interneural communication involves diffusion of certain chemicals across synapses. Memory is likewise a chemical phenomenon, involving chemical changes in nerve cells and associated structures. Regardless of these physical differences, however, the two systems possess a fundamental logical similarity, as we've seen. Is it now possible to map the information flow in the human system in the same way that we are accustomed to in electronics?

Yes, to a certain extent. First of all, no one alive and in good health would consent to have men in white lab coats probing about inside his or her head. So direct information about the living, functioning nervous system is hard to come by. But a lot of data can be obtained concerning brain structure in an indirect way, and this is the domain of cognitive psychology. Research along these lines can be frustrating; it's very much like trying to figure out what's inside a computer by seeing how the machine responds to different sequences of input signals. Nevertheless, psychological data, together with whatever biologists can provide, has allowed many deductions about the brain's organizations.

Let's look at Figure 4, which is a simplified human cognitive map. Memory elements are represented by squares, while circles stand for control processes. This is an arbitrary distinction because both functions, memory and control, are apparently performed by cells. Keeping that fact in mind, let's analyze the major features of Figure 4, saving the details for later. First, visual and auditory information from the outside world are converted by the eyes and ears respectively into nervous impulses. These signals are routed to buffer memories, the sensory registers, which act to preserve fleeting data long enough for the rest of the system to process it. By convention the visual and auditory sensory registers are called, respectively, the "icon" and the "echo."

The block labelled LTM is long-term memory, a presumably permanent storehouse filled with all the important facts accumulated during a lifetime. For example, LTM contains enormous quantities of information such as 1+1=2, V=IR, the rules of English grammar, telephone numbers, and so on. In contrast, STM or short-term memory has a very limited capacity. Furthermore, information fades away or gets dislodged very easily from STM. It is possible to consciously retain STM information, however, by the process of



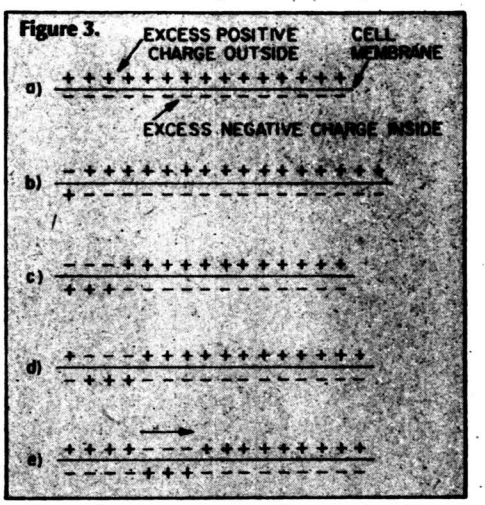
A motor neuron, one of the basic building blocks of the biological information processing system, is a single cell and functions as a multi-input majority gate. The inputs to the dendrites and the cell body come from other neurons. These inputs, often numbering in the hundreds, are averaged and the majority input determines the output along the axon—sort of biological democracy. The axon then leads to another neuron.

rehearsal, repeating things mentally to one's self. As the arrows of Figure 4 indicate, rehearsal takes information out of STM and then re-enters it. At the same time, rehearsal can facilitate entry of information into LTM, as we'll see later.

Recognition. The final circle represents two independent processes, chunking and pattern recognition. Because these processes seem to have the same inputs and output, they coincide on the map. Chunking is an organizational process whereby sensory register information gets grouped by means of LTM rules before entering STM. This chunking does two important things: 1) it allows more information to be crammed into STM, and 2) it makes the information easier for LTM to assimilate. Note that information cannot enter LTM directly; instead, it must be worked on in STM first. For this reason, STM is commonly called working memory by psychologists. Apparently, STM is the site of what we experience as consciousness. The last process, pattern recognition, is the identification of visual or auditory inputs, and is a very important part of conscious experience

As we stated earlier, the map in Figure 4 is far from complete. Missing entirely are the autonomic processes, such as breathing, which the brain controls. Gone too are the ouputs which, for example, would control muscular action. It would also appear from Figure 4 that all inputs are processed equally, yet we kow this is not true. As you read these lines countless sounds, sights, and pressure sensations are available to your senses, yet your attention is fixed on the letters that you scan. This phenomenon of selective attention is extremely important, but a little too complex to cover here.

Keeping in mind the fact that our cognitive map is greatly simplified, let's consider its features in more detail, starting with the sensory registers. Experiments with the icon have revealed that visual information is stored for less than a second before fading. Moreover, new visual stimuli erase the old iconic information. If this did not happen, our apprehension of an image would lag about a second behind its appearance. By contrast, echoic storage lasts longer than that of the icon (the exact duration is subject to dispute, however), and erasure per se doesn't seem to occur. The necessity of echoic storage is readily seen when you consider that language is a serial phenomenon, made up of sounds which follow each other in time. To make sense out of language, what is and what was must both be available. For example, a single word like *compiter* is made up of many basic sounds called phonemes. All must be available in order for the pattern recognition process to identify the word.



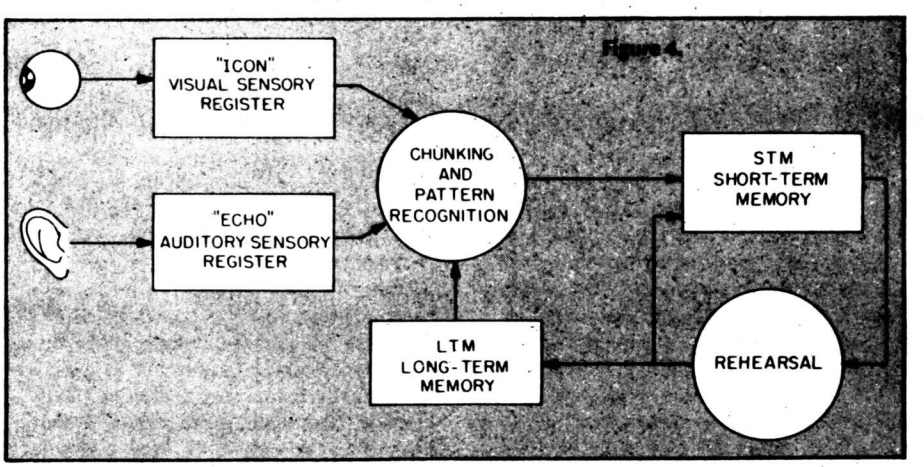
Nerve signals are waves of temporary polarity reversal along the cell membane. In the first drawing above (a) the nerve membrane is in a quiescent state, in (b) the polarity reversal begins and progresses along the membrane (c & d). The polarity returns to normal (e) and this return travels behind the reversal at an equal speed (about 100 mps).

STM. Let's consider STM. One piece of evidence suggesting an STM that is distinct from LTM is Milner's Syndrome, a mental impairment caused by damage to the hippocampal area of the brain. Victims of this syndrome are quite normal in most respects, but are unable to permanently memorize new information. For example, given a list of words, they can retain the list in memory for any desired length of time by continually rehearsing. When instructed to stop rehearsing, they lose the list in a matter of a minute or so. One patient was able to read the same magazine again and again without getting bored. The patient's doctor had to introduce himself daily, even though he had been treating the man for years. Yet, the patient could recall the events in his life before the accident. According to our cognitive map, we can theorize that the damage broke the link between LTM and STM. The victim could use STM and recall old items from LTM, but he was unable to put new information into long-term storage.

Not only is STM storage temporary but it also possesses only limited capacity—typically five to nine items. This figure varies according to the nature of the items being memorized, but the average person's short-term capacity is about seven letters, numbers, or words. This seven-item limit is commonly referred to as the memory span. Let's clarify the memory span concept now. If a list of items is presented one at a time to a subject, that person can recall the list in perfect order, after seeing the items just once, if the list length is less than or equal to about seven items. Mistakes will appear with longer lists. Of course, we can memorize more than seven items, but to do so will require several presentations of the list plus some rehearsing. Since rehearsal both refreshes STM and transfers information to LTM, learning of lists longer than the memory span evidently requires storage space in LTM.

Suppose that the following list of

(Continued on page 80)



A simplified model of the information processing system of humans and higher animals. In addition to the "ECHO" and "ICON" sensory registers there are others for taste, smell and touch. In the lower box, the LTM, is stored everything you know except for the most recent inputs which are being processed as you read this page. These immediate inputs are going into the short term memory and rehersal loop; some of what you have read will be recorded in the LTM—there is, however, no guarantee your brain will be able to recall it. Something may be in memory but the brain can't always find it.

CHESSBOARD RUMBLE

HCH squares off three chess-playing computer combatants!

by Norm Myers

FEW DEVELOPMENTS HAVE struck the imagination of hobbyists and engineers like microprocessors—and few applications have dazzled hobbyists, engineers, and the public in general more than the new microcomputer chess games. These units actually play chess and help teach you to play the game. The whole idea of computerized chess is so dependent upon high-powered microprocessor chips and high-speed circuitry, as well as other computer-like design attributes that the whole idea of a small computerized chess game was impossible only a few years ago. But now, with the availability of small, inexpensive microprocessor chips, and other electronic devices the game is a reality.

Now we will do something never done before—as far as we know. We have located and tried three different computer chess games. In this brief space we will discuss the basics of each and then—hold your seat—we will play two of those computers against each other.

Let's Meet the Players. All three units are becoming available in department stores; their prices are shown herein along with the manufacturers' addresses.

First there is our old friend the *Chess Challenger* with the four character LED display, three levels of play (although it is available as a basic model with one level), and a response time ranging from a few seconds to about one minute. It will play either black or white, as you wish. If you want to set up pieces on the board in a particular way so as to try the computer against, for example, the Sunday *New York Times* chess problem, you can do it. You can verify the location of pieces on the board by pressing a key that causes a read-out for each square.

Then there is *CompuChess* which also has a four character LED display, but it has six levels of play with the first four requiring from a few seconds to 15 minutes for average players, and the next two levels requiring up to 18 and 48 hours, respectively, for super players and problem crunching. The level can be changed at any time during the game so you can start with low levels, then move up in level as the pieces thin out so its response time is not quite so long. CompuChess will play either black or white pieces and will let you set up the pieces on the board in any way you wish in case you want to try a specific situation.

Finally, there is *Boris*. This Russian-named unit is a bit different from the other two. Besides playing either black or white, it will play against itself at any point during a game. So if it is your move, and you want to see what Boris would recommend, just let it make the move for you. The display is also different. Instead of a four character LED display where each character can only be made from the form of a block figure eight, there are eight characters and each can display a whole range of strange looking symbols and characters. The result is that Boris, when asked, will show a whole row across the board with the King's knight on one square, the queen on some other, etc. A knight actually looks like a stick horse on this display, and a queen looks like a Miss America crown. It's easy to correct a dumb move you just made, or to set up the pieces in a certain way on the board. Simply call up row One, put pieces where you want them and go on to row Two, etc. This can be done anytime during a game. Boris is a *kibitzer*, he *loves* to comment on your moves. He may spell out "I expected that" along the display or "Are you ranked?" or some other friendly remark.

Boris also has a different approach to level of play. Instead of selecting one of the several levels at the beginning of the game, you select the amount of time you will allow Boris to think. This can be any duration from a few seconds to 100 hours. The duration can be changed at any time during the game. The designers claim that at 100 hours for each move Boris will give a chess master a callenging game, but I

and input/output (I/O) ports. The user assembles the basic computer and then installs the amount of memory and I/Os needed for his specific applications. Both companies provide inexpensive software, particularly elementary and advanced (extended) versions of BASIC, a high level interactive language that is just about standard for all hobby computers, and virtually a standard for the larger personal computers used in business applications.

Basic Training. Since BASIC is so important to hobby computers, let's digress and take a look at what is meant by BASIC. While it's fun to use a trainer and program by moving an electron here, another electron there, it is quite tiring to program by moving *bits* which represent a binary 1 or 0. In fact, make one single error in bit programming and an hour or so of effort goes down the drain. In truth, simply adding two numbers through a binary bit program is difficult; multiplying two numbers is next to impossible. In the higher level trainers, the bits are represented by an *op code*, or assembly language, and the user can instruct the computer by simply entering two numbers. For example, the numerals 56 might tell the computer to store the following numerals in a register; 65 might mean the next number is stored in another register, and 83 might tell the computer to add the stored numbers. In assembly language, you might actually use a command such as ADDA to enter a number in a register.

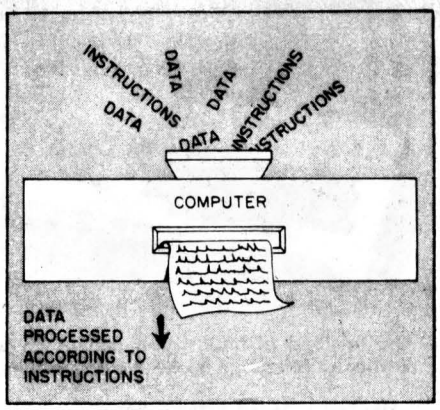
Much programming can be, and is done, in assembly language, but it is extremely difficult and time consuming for the average hobbyist.

How much easier it would be to simply communicate with your computer in plain English. For example, if you wanted the result of 2 times 2 (2*2), the easiest thing to do would be to type out on a terminal PRINT 2*2, and the computer would respond by printing out a 4 on a TV screen or printer. Well, this is exactly what's done if the computer is programmed with a high level language, or HLL, as it is called in many computer books.

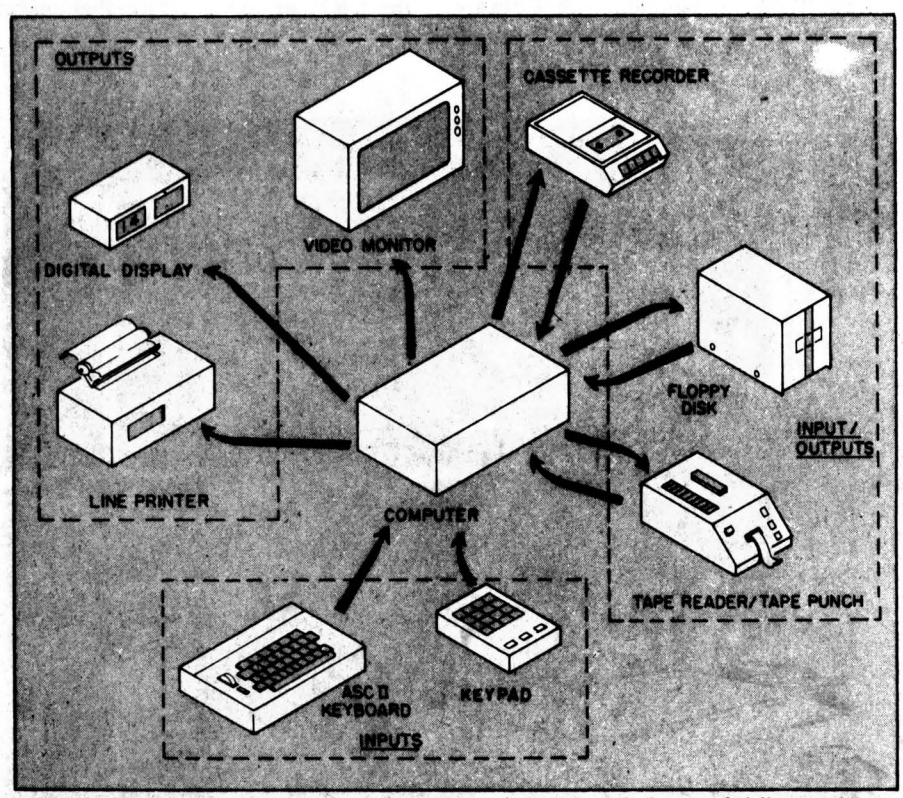
High Level Language. There are several popular high level languages, most originally intended for specific computers such as those of IBM and DEC (Digital Equipment Corporation), or for specific applications. Some higher level languages are almost as difficult to learn as a foreign language such as French or German. One of the scientific computer languages which is somewhat difficult to learn is Fortran. Partially to assist college students in learning Fortran, two of the Dartmount University staff developed a language called BASIC—an acronym for Beginner's All-purpose Symbolic Instruction Code.

BASIC turned out to be more than a way to learn Fortran. Using familiar English words to perform standardized computer and math operations, BASIC soon became a virtual "standard" language from grade school to college and on through commercial applications. (Today, you'll find business applications which were formerly written in Cobol —a business oriented HLL—being written in BASIC.) Literally millions of students were trained in BASIC, primarily because it is easy to learn, uses standard teletypewriter symbols, which are almost an exact duplicate of standard typewriter symbols, and because it can do so many things. Fact is, many functions not written into the original BASIC have been added, and new functions are continuously added. In short, BASIC is a growing computer language.

It was therefore predestined that BASIC would be the standard hobby computer language. A few early, feeble attempts resulted in a rather limited BASIC that did not process decimals—it was an *integer* BASIC, rather useless for most grade and high school applications. You can't even keep a checkbook



A Computer takes data and handles it according to your instructions. You may tell the computer to multiply two words of data, or to read a list and record all the words with a certain combination of letters. The computer cannot think, it can only process data according to instructions.



Input/Output devices come in a wide variety of shapes and sizes and fall into three major categories. First is the straight Input device such as a keyboard, or keypad which converts letters and numbers into series of electronic signals that can be understood by the computer. The Outputs present these signals, after they have been processed, into recognizable letters and numbers. The two-way Input/Output devices can put data in to the computer and they can take data from the computer and store it for later reuse.

have found it to be quite challenging at ten seconds. As with the other computer units, Boris looks ahead a few moves; the point of the time is to let it look ahead as far as it can in the time allotted.

One interesting pitfall is that computer units do not want to let go if the software algorithms indicate that there is an advantage to a certain move or set of moves. This causes trouble when, for example, a piece of yours is locked into a corner and you can only move it back and forth in this endless cycle with no way to break it. The cycles can be complicated, with many moves, or just simple ones. In a person-to-person chess game, someone has to give or a stalemate is declared. We have found that of the pitfalls listed above, our Boris unit appeared to be programmed to avoid the obvious mistakes, but actually all three units provide a very challenging chess game.

Playing one computer chess unit against the other turned out to be very interesting because each unit planned ahead a couple of moves so we could see each computer setting traps for the other.

Onward, To the Battlefield! We pitted Boris against Chess Challenger for three games and CompuChess against Chess Challenger for three games. The first game of each three-game set was at the lowest level on each unit with one playing white (starting the game), the second game with the other playing white, and the third on level 2 (around 10-15 seconds per move). The result was a lot of intriguing chess—all tied up like a novel with lots of little side plots, pitfalls, defenses, and threats.

The computers do, however, appear to behave differently. Boris will shoot one of its phrases across the screen that adds humor to the game; he also appears very willing to exchange queens rather than running away if they are attacking each other. This strategy simplifies the board a lot and probably gives Boris an edge over having to compute moves with the queens because it can make more trial moves in its microprocessor before time is up. So Boris comes on as somewhat aggressive, and it is designed to have different opening moves to add variety to the games.

CompuChess also has variety in its moves, not only at the beginning but throughout the game. If two moves are of nearly equal value, it will choose one and during another game it may choose the other. It appears to think carefully through every possibility before making a move, which gives it a defensive posture during a game but when an opportunity arises to gain an opponent's piece it will set up its moves to do just that. On level one it makes every possible move in its memory and looks at the opponent's possible responses before selecting the best move. On level two it will look at its possible responses to the opponent's response—and so on for the other levels. Boris and Chess Challenger are programmed in a similar manner because it is impossible to build a full array of pattern recognition solutions (Ah! This is pattern R, so I'll set up my Sicilian defense!) in a microcomputer. The only way is to have the computer try to move every piece and then to evaluate what will happen.

Chess Challenger appears to have a little less variety in its play than the CompuChess unit and that may be because it rarely ends up evaluating two situations as equal and therefore almost never "flips a coin" to decide which move to make. It uses a full 16 bit register to score evaluations, giving it a resolution of one part in 216 (64,000), so it rarely calls moves equal in value. From an opponent's viewpoint this means the Chess Challenger will not give you the slightest break—it always does its level best.

Chess on a microcomputer? It's here now, all three of these units giving a

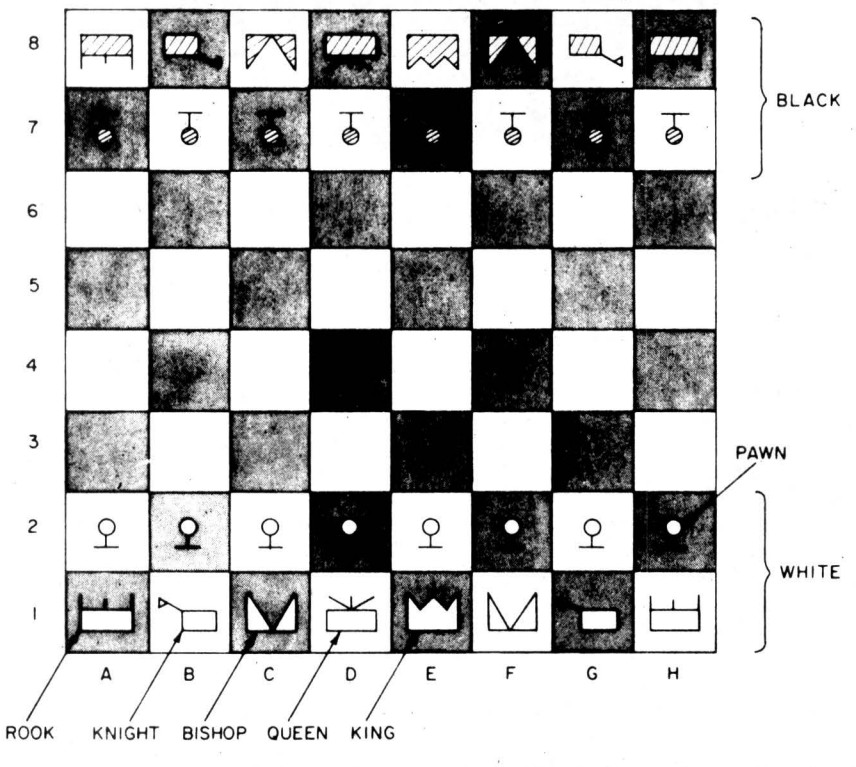
(Continued on page 82)

A COMPUTER REYKJAVIK

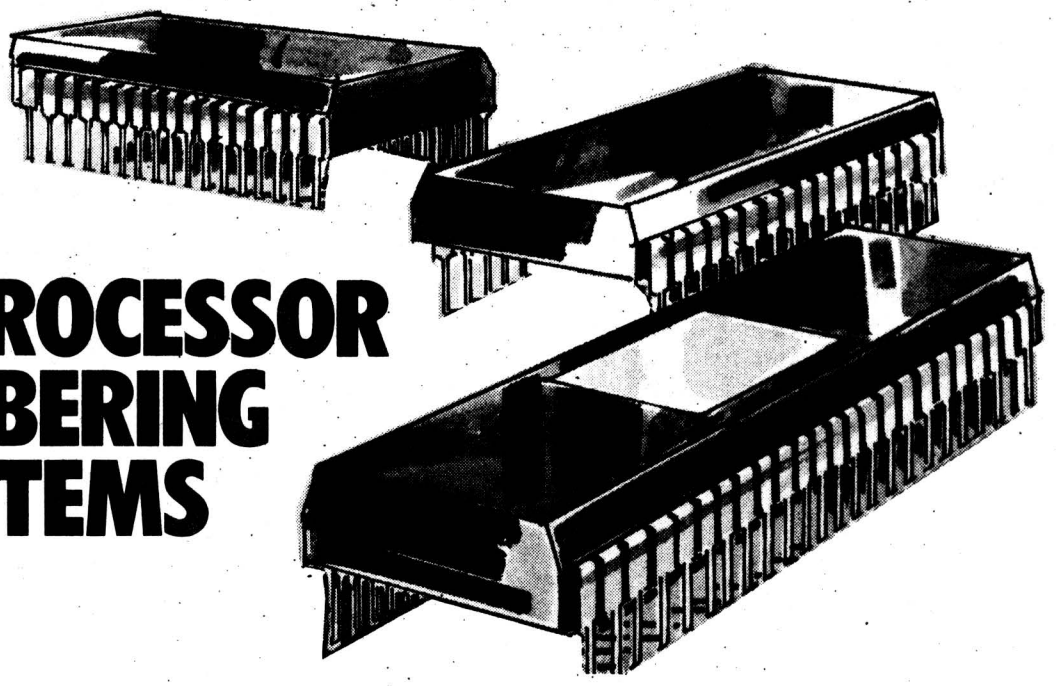
As for the games, sometimes one computer would really whip its challenging computer one game, only to be defeated on the next game. The games were not duplicates of each other. One small difference in a move would turn a whole game around. Let's look at the beginning of a Boris (white) versus Chess Challenger (black) game. (E2E3 means move the piece at E2 to E3).

White	Black	
E2E3	D7D5	Advanced openings being used by both units.
F1D3	G8F6	
B1C3	C8G4	Challenger attacks queen. "Very interesting," says Boris.
F2F3	G4G7	Challenger retreats from its attack. Boris opens side to breathe a little.
B2B4	E7E4	
A1B1	B8C6	Challenger attacking white pawn.
C1A3	F8E7	Boris protects that pawn with bishop.

You can see the intrigue—attack, retreat, try again somewhere else, etc. This particular game lasted about two hours and required over one hundred moves by each computer. In the end Boris won, but it was a close game with only five white pieces and four black pieces left.



Shown here with the symbols used by the Boris machine is the starting position for all games of chess. White moves first. Object is to trap your opponent's king before he can trap yours. Boris also has a set of wisecracks he will often spout during the game!



MICROPROCESSOR NUMBERING SYSTEMS

What you will learn. To understand and work with microprocessors, it's necessary to understand the binary numbers and codes which make up the language of these microprocessors. Approach the problem logically, and you will find that it's all even easier than you might suspect.

Here we condense the beginning of the *Heathkit* home-study course on microcomputers. We think you will agree that this course makes learning interesting and fun.

First, you will learn about numbering systems in general because microprocessors *think* in numbers. Given any decimal number, you will be able to convert it into its binary or octal equivalent. You'll be able to convert back and forth from binary to octal. You will know what a *radix* is, an integer, and a whole lot more.

Once you learn these basics, we will guide you into the world of hexadecimal numbers and show you why they are so important to understand if you intend to work with binary-based circuitry. By the time we finish, you will be swiftly converting back and forth between the binary and hexadecimal systems.

Finally, you'll learn a bit about binary codes. You will discover the secret of the Binary Coded Decimal and how to convert between that code and the decimal system. You'll discover the Gray code and then you'll meet the modern-day ASCII code and its ancestor, the BAUDOT code.

Follow along and join us in understanding microprocessors!

DECIMAL NUMBER SYSTEM

The number system we are all familiar with is the decimal number system. This system was originally devised by Hindu mathematicians in India about 400 A.D. The Arabs began to use the system about 800 A.D., where it became known as the Arabic Number System. After it was introduced to the European community about 1200 A.D., the system soon acquired the title "decimal number system."

A basic distinguishing feature of a number system is its **base** or **radix**. The base indicates the number of characters or digits used to represent quantities in that number system. The decimal number system has a base or radix of 10 because we use the ten digits 0 through 9 to represent quantities. When a number system is used where the base is not known, a subscript is used to show the base. For example, the number 4603_{10} is derived from a number system with a base of 10.

Positional Notation. The decimal number system is positional or weighted. This means each digit position in a number carries a particular weight which determines the magnitude of that number. Each position has a weight de-

Figure 1

$10^0 = 1$
$10^1 = 10$
$10^2 = 100$
$10^3 = 1{,}000$
$10^4 = 10{,}000$
$10^5 = 100{,}000$
$10^6 = 1{,}000{,}000$
$10^7 = 10{,}000{,}000$
$10^8 = 100{,}000{,}000$
$10^9 = 1{,}000{,}000{,}000$

termined by some power of the number system base, in this case 10. The positional weights are 10^0 (units) (1 unit), 10^1 (tens), 10^2 (hundreds), etc. Refer to Figure 1 for a condensed listing of powers of 10.

We evaluate the total quantity of a number by considering the specific digits and the weights of their positions. For example, the decimal number 4603 is written in the shorthand notation with which we are all familiar. This number can also be expressed with positional notation.

$(4 \times 10^3) + (6 \times 10^2) + (0 \times 10^1) + (3 \times 10^0) =$
$(4 \times 1000) + (6 \times 100) + (0 \times 10) + (3 \times 1) =$
$4000 + 600 + 0 + 3 = 4603_{10}$

To determine the value of a number, multiply each digit by the weight of its position and add the results.

Fractional Numbers. So far, only *integer* or whole numbers have been discussed. An integer is any of the natural numbers, the negatives of these numbers, or zero (that is, 0, 1, 4, 7, etc.). Thus, an integer represents a whole or complete number. But, it is often necessary to express quantities in terms of fractional parts of a whole number.

Decimal fractions are numbers whose positions have weights that are negative powers of ten such as $10^{-10} = 10^{-10} = 1/10 = 0.1$, $10^{-2} = 1/100 = 0.01$, etc.

Figure 2 provides a condensed listing of negative powers of 10 (decimal fractions).

Figure 2

$10^{-1} = 1/10$	$= 0.1$
$10^{-2} = 1/100$	$= 0.01$
$10^{-3} = 1/1000$	$= 0.001$
$10^{-4} = 1/10{,}000$	$= 0.0001$
$10^{-5} = 1/100{,}000$	$= 0.00001$
$10^{-6} = 1/1{,}000{,}000$	$= 0.000001$

A radix point (decimal point for base 10 numbers) **separates** the **integer** and **fractional** parts of a number. The integer or whole portion is to the left of the decimal point and has positional weights of units, tens, hundreds, etc. The fractional part of the number is to the right of the decimal point and has positional weights of tenths, hundredths, thousandths, etc. To illustrate this, the decimal number 278.94 can be written with positional notation as shown below.

$(2 \times 10^2) + (7 \times 10^1) + (8 \times 10^0) \times (9 \times 10^{-1} +$
$(4 \times 10^{-2}) = (2 \times 100) + (7 \times 10) + (8 \times 1) +$
$(9 \times 1/10) + (4 \times 1/100) = 200 + 70 + 8 + 0.9 +$
$0.04 = 278.94_{10}$

In this exemple, the left-most digit (2×10^2) is the **most significant digit** or MSD because it carries the greatest weight in determining the value of the number. The right-most digit, called the **least significant digit** or LSD, has the lowest weight in determining the value of the number. Therefore, as the term implies, the MSD is the digit that will affect the greatest change when its value is altered. The LSD has the smallest effect on the complete number value.

BINARY NUMBER SYSTEM

The simplest number system that uses positional notation is the binary number system. As the name implies, a **binary** system contains only two elements or states. In a number system this is expressed as a base of 2, using the digits 0 and 1. These two digits have the same basic value as 0 and 1 in the decimal number system.

Because of its simplicity, microprocessors use the binary number system to manipulate data. Binary data is represented by binary digits called **bits.** The term bit is derived from the contraction of **b**inary dig**it**. Microprocessors operate on groups of bits which are referred to as words. The binary number 11101101 contains eight bits.

Positional Notation

As with the decimal number system, each bit (digit) position of a binary number carries a particular weight which determines the magnitude of that number. The weight of each position is determined by some power of the number system base (in this example 2). To evaluate the total quantity of a number, consider the specific bits and the weights of their positions. (Refer to Figure 3 for a condensed listing of powers of 2.) For example, the binary number 110101 can be written with positional notation as follows:

$(1 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

To determine the decimal value of the binary number 110101, multiply each bit by its positional weight and add the results.

$(1 \times 32) + (1 \times 16) + (0 \times 8) + (1 \times 4) + (0 \times 2) + (1 \times 1) =$
$32 + 16 + 0 + 4 + 0 + 1 = 53_{10}$

Figure 3

$2^0 = 1_{10}$	$2^6 = 64_{10}$
$2^1 = 2_{10}$	$2^7 = 128_{10}$
$2^2 = 4_{10}$	$2^8 = 256_{10}$
$2^3 = 8_{10}$	$2^9 = 512_{10}$
$2^4 = 16_{10}$	$2^{10} = 1024_{10}$
$2^5 = 32_{10}$	$2^{11} = 2048_{10}$

Fractional binary numbers are expresesd as negative powers of 2. Figure 4 provides a condensed listing of negative powers of 2. In positional notation, the binary number 0.1101 can be expressed as follows:

$$(1 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4})$$

To determine the decimal value of the binary number 0.1101, multiply each bit by its positional weight and add the results.

$(1 \times 1/2) + (1 \times 1/4) + (0 \times 1/8) + (1 \times 1/16) =$
$0.5 + 0.25 + 0 + 0.0625 = 0.8125_{10}$

In binary number system, radix point is called binary point.

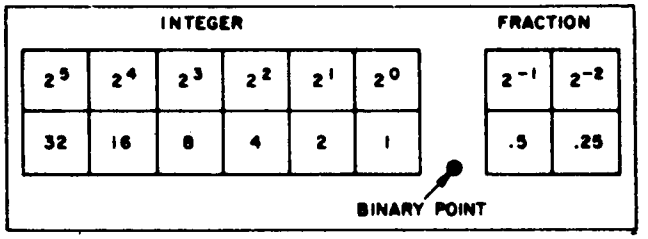
Figure 4

$2^{-1} = 1/2$	$= 0.5_{10}$
$2^{-2} = 1/4$	$= 0.25_{10}$
$2^{-3} = 1/8$	$= 0.125_{10}$
$2^{-4} = 1/16$	$= 0.625_{10}$
$2^{-5} = 1/32$	$= 0.03125_{10}$
$2^{-6} = 1/64$	$= 0.015625_{10}$
$2^{-7} = 1/128$	$= 0.0078125_{10}$
$2^{-8} = 1/256$	$= 0.00390625_{10}$

Converting Between the Binary and Decimal Number Systems

In working with microprocessors, you will often need to determine the decimal value of binary numbers. In addition, you will find it necessary to convert a specific decimal number into its binary equivalent. The following information shows how such conversions are accomplished.

Binary to Decimal. To convert a binary number into its decimal equivalent, add together the weights of the positions in the number where binary 1's occur. The weights of the integer and fractional positions are indicated below.



As an example, convert the binary number 1010 into its decimal equivalent. Since no binary point is shown, the number is assumed to be an integer number, where the binary point is to the right of the number. The right-most bit, called the **least significant bit** or LSB, has the lowest integer weight of $2^0=1$. The left-most bit is the **most significant bit** (MSB) because it carries the greatest weight in determining the value of the number. In this example, it has a weight of $2^3=8$. To evalaute the number, add together the weights of the positions where binary 1's appear. In this example, 1's occur in the 2^3 and 2^1 positions. The decimal equivalent is ten.

Binary Number	1	0	1	0_2
Position Weights	2^3	2^2	2^1	2^0

Decimal Equivalent

$$8+0+2+0=10_{10}$$

To further illustrate this process, convert the binary number 101101.11 into its decimal equivalent.

Binary Number	1	0	1	1	0	1	.1	1
Position Weights	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}

Decimal Equivalent

$$32+0+8+4+0+1+.5+.25=45.75_{10}$$

Decimal to Binary. A decimal integer number can be converted to a different base or radix through successive divisions by the desired base. To convert a decimal integer number to its binary equivalent, successively divide the number by 2 and note the remainders. When you divide by 2, the remainder will always be 1 or 0. The remainders form the equivalent binary number.

As an example, the decimal number 25 is converted into its binary equivalent.

$25\div2=12$ with remainder	1 ← LSB
$12\div2=6$	0
$6\div2=3$	0
$3\div2=1$	1
$1\div2=0$	1 ← MSB

Divide the decimal number by 2 and note the remainder. Then divide the quotient by 2 and again note the remainder. Then divide the quotient by 2 and once again note the remainder. Continue this division process until 0 results. Then collect remainders beginning with the last or most significant bit (MSB) and proceed to the first or least significant bit (LSB). The number $11001_2=25_{10}$. Notice that the remainders are collected in the reverse order. That is, the first remainder becomes the least significant bit, while the last remainder becomes the most significant bit.

Note: Do not attempt to use a calculator to perform this conversion. It would only supply you with confusing results.

To further illustrate this, the decimal number 175 is converted into its binary equivalent.

$175\div2=87$ with remainder	1 ← LSB
$87\div2=43$	1
$43\div2=21$	1
$21\div2=10$	1
$10\div2=5$	0
$5\div2=2$	1
$2\div2=1$	0
$1\div2=0$	1 ← MSB

The division process continues until 0 results. The remainders are collected to produce the number $10101111_2=175_{10}$.

To convert a decimal fraction to a different base or radix, multiply the fraction successively by the desired base and record any integers produced by the multiplication as an overflow. For example, to convert the decimal fraction 0.3125 into its binary equivalent, multiply repeatedly by 2.

$0.3125 \times 2=0.625=0.625$ with overflow	0 ← MSB
$0.6250 \times 2=1.250=0.250$	1
$0.2500 \times 2=0.500=0.500$	0
$0.5000 \times 2=1.000=0$	1 ← LSB

These multiplications will result in numbers with a 1 or 0 in the units position (the position to the left of the decimal point). By recording the value of the units position, you can construct the equivalent binary fraction. This units position value is called the "overflow." Therefore, when 0.3125 is multiplied by 2, the overflow is 0. This becomes the most significant bit (MSB) of the binary equivalent fraction. Then 0.625 is multiplied by 2. Since the product is 1.25, the overflow is 1. When there is an overflow of 1, it is effectively subtracted from the product when the value is recorded. Therefore, only 0.25 is multiplied by 2 in the next multiplication process. This method continues until an overflow with no fraction results. It is important to note that you can not always obtain 0 when you multiply by 2. Therefore, you should only continue the conversion process to the accuracy or precision you desire. Collect the conversion overflows beginning at the radix (binary) point with the MSB and proceed to the LSB. This is the same order in which the overflows were produced. The number $0.0101_2=0.3125_{10}$.

To further illustrate this process, the decimal fraction 0.84375 is converted into its binary equivalent.

$0.90625 \times 2=1.8125=0.8125$ with overflow	1 ← MSB
$0.81250 \times 2=1.6250=0.6250$	1
$0.62500 \times 2=1.2500=0.2500$	1
$0.25000 \times 2=0.5000=0.5000$	0
$0.50000 \times 2=1.0000=0$	1 ← LSB

The multiplication process continues until either 0 or the desired precision is obtained. The overflows are then collected beginning with the MSB at the binary (radix) point and proceeding to the LSB. The number $0.11101_2 = 0.90625_{10}$.

If the decimal number contains both an integer and fraction, you must separate the integer and fraction using the decimal point as the break point. Then perform the appropriate conversion process on each number portion. After you convert the binary integer and binary fraction, recombine them. For example, the decimal number 14.375 is converted into its binary equivalent.

$14.375_{10} = 14_{10} + 0.375_{10}$

$14 \div 2 = 7$	with remainder	$0 \leftarrow$ LSB
$7 \div 2 = 3$		1
$3 \div 2 = 1$		1
$1 \div 2 = 0$		$1 \leftarrow$ MSB

$14_{10} = 1110_2$

$0.375 \times 2 = 0.75 = 0.75$ with overflow	$0 \leftarrow$ MSB
$0.750 \times 2 = 1.50 = 0.50$	1
$0.500 \times 2 = 1.00 = 0$	$1 \leftarrow$ LSB

$0.375_{10} = 0.011_2$

$14.375_{10} = 14_{10} + 0.375_{10} = 1110_2 + 0.011_2 = 1110.011_2$.

OCTAL NUMBER SYSTEM

Octal is another number system that is often used with microprocessors. It has a base (radix) of 8, and uses the digits 0 through 7. These eight digits have the same basic value as the digits 0–7 in the decimal number system.

As with the binary number system, each digit position of an octal number carries a positional weight which determines the magnitude of that number. The weight of each position is determined by some power of the number system base (in this example, 8). To evaluate the total quantity of a numbr, consider the specific digits and the weights of their positions. Refer to Figure 5 for a condensed listing of powers of 8. For example, the octal number 372.01 can be written with positional notation as follows:

$$(3 \times 8^2)+(7 \times 8^1)+(2 \times 8^0)+(0 \times 8^{-1})+(1 \times 8^{-2})$$

Figure 5

$8^{-4} = 1/4096 = 0.000244140625_{10}$
$8^{-3} = 1/512 = 0.001953125_{10}$
$8^{-2} = 1/64 = 0.015625_{10}$
$8^{-1} = 1/8 = 0.125_{10}$
$1_{10} = 8^0$
$8_{10} = 8^1$
$64_{10} = 8^2$
$512_{10} = 8^3$
$4096_{10} = 8^4$
$32768_{10} = 8^5$
$262144_{10} = 8^6$

The decimal value of the octal number 372.01 is determined by multiplying each digit by its positional weight and adding the results. As with decimal and binary numbers, the radix (octal) point separates the integer from the fractional part of the number.

$(3 \times 64)+(7 \times 8)+(2 \times 1)+(0 \times 0.125)+(1 \times 0.015625) = 192+56+2+0+0.015625 = 250.015625_{10}$

Conversion From Decimal to Octal

Decimal to octal conversion is accomplished in the same manner as decimal to binary, with one exception; the base number is now 8 rather than 2. As an example, the decimal number 194 is converted into its octal equivalent.

$194 \div 8 = 24$	with remainder	$2 \leftarrow$ LSD
$24 \div 8 = 3$		0
$3 \div 8 = 0$		$3 \leftarrow$ MSD

Divide the decimal by 8 and note the remainder. (The remainder can be any number from 0 to 7.)

Then divide the quotient by 8 and again note the remainder. Continue dividing until 0 results. Finally, collect the remainders beginning with the last or most significant digit (MSD) and proceed to the first or least significant digit (LSD). The number $302_8 = 194_{10}$. Figure 6 illustrates the relationship between the first several decimal, octal, and binary integers.

To further illustrate this process, the decimal number 175 is converted into its octal equivalent.

$175 \div 8 = 21$	with remainder	$7 \leftarrow$ LSD
$21 \div 8 = 2$		5
$2 \div 8 = 0$		$2 \leftarrow$ MSD

Figure 6

DECIMAL	OCTAL	BINARY
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	10	1000
9	11	1001
10	12	1010
11	13	1011
12	14	1100
13	15	1101
14	16	1110
15	17	1111
16	20	10000
17	21	10001
18	22	10010
19	23	10011
20	24	10100

The division process continues until a quotient of 0 results. The remainders are collected, producing the number $257_8 = 175_{10}$.

To convert a decimal fraction to an octal fraction, mul-

tiply the fraction successively by 8 (octal base). As an example, the decimal fraction 0.46875 is converted into its octal equivalent.

$0.46875 \times 8 = 3.75 = 0.75$ with overflow	3 ← MSD
$0.75000 \times 8 = 6.00 = 0$	6 ← LSD

Multiply the decimal number by 8. If the product exceeds one, subtract the integer (overflow) from the product. Then multiply the product fraction by 8 and again note any "overflow." Continue multiplying until an overflow, with 0 for a fraction, results. Remember, you can not always obtain 0 when you multiply by 8. Therefore, you should only continue this conversion process to the accuracy or precision you desire. Collect the conversion overflows beginning at the radix (octal point) with the MSD and proceed to the LSD. The number $0.36_8 = 0.46875_{10}$. Figure 7 illustrates the relationship between decimal, octal, and binary fractions.

Now, the decimal fraction 0.136 will be converted into its octal equivalent with four-place precision.

$0.136 \times 8 = 1.088 = 0.088$ with overflow	1 ← MSD
$0.088 \times 8 = 0.704 = 0.704$	0
$0.704 \times 8 = 5.632 = 0.632$	5
$0.632 \times 8 = 5.056 = 0.056$	5 ← LSD

$0.136_{10} = 0.1055_8$

The number 0.1055_8 approximately equals 0.136_{10}. If you convert 0.1055_8 back to decimal (using positioned notation) you will find $01055_8 = 0.135986328125_{10}$. This example shows that extending the precision of your conversion is of little value unless extreme accuracy is required.

As with decimal to binary conversion of a number that contains both an integer and fraction, decimal to octal conversion requires two operations. You must separate the integer from the fraction, then perform the appropriate conversion on each number. After you convert them, you must recombine the octal integer and octal fraction. For example, convert the decimal number 124.78125 into its octal equivalent.

Figure 7

DECIMAL	OCTAL	BINARY
0.015625	0.01	0.000001
0.03125	0.02	0.00001
0.046875	0.03	0.000011
0.0625	0.04	0.0001
0.078125	0.05	0.000101
0.09375	0.06	0.00011
0.109375	0.07	0.000111
0.125	0.1	0.001
0.140625	0.11	0.001001
0.15625	0.12	0.00101
0.171875	0.13	0.001011
0.1875	0.14	0.0011
0.203125	0.15	0.001101
0.21875	0.16	0.00111
0.234375	0.17	0.001111
0.25	0.2	0.01
0.265625	0.21	0.010001
0.28125	0.22	0.01001
0.296875	0.23	0.010011
0.3125	0.24	0.0101

$124.78125_{10} = 124_{10} + 0.78125_{10}$

$124 \div 8 = 15$ with remainder	4 ← LSD
$15 \div 8 = 1$	7
$1 \div 8 = 0$	1 ← MSD

$124_{10} = 174_8$

$0.78125 \times 8 = 6.25 = 0.25$ with overflow	6 ← MSD
$0.25000 \times 8 = 2.00 = 0$	2 ← LSD

$0.78125_{10} = 0.62_8$

$124.78125_{10} = 124_{10} + 0.78125_{10} = 174_8 + 0.62_8 = 174.62$

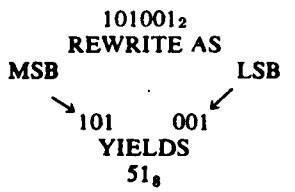
Converting Between the Octal and Binary Number Systems

Microprocessors manipulate data using the binary number system. However, when larger quantities are involved, the binary number system can become cumbersome. Therefore, other number systems are frequently used as a form of binary shorthand to speed-up and simplify data entry and display. The octal number system is one of the systems that is used in this manner. It is similar to the decimal number system, which makes it easier to understand numerical values. In addition, conversion between binary and octal is readily accomplished because of the value structure of octal. Figures 6 and 7 illustrate the relationship between octal and binary integers and fractions.

As you know, three bits of binary number exactly equal eight value combinations. Therefore, you can represent a 3-bit binary number with a 1-digit octal number.

$101_2 = (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 4 + 0 + 1 = 5_8$

Because of this relationship, converting binary to octal is simple and straight forward. For example, binary number 101001 is converted into its octal equivalent.



To convert a binary number to octal, first separate the number into groups containing three bits, beginning with the least significant bit. Then convert each 3-bit group into its octal equivalent. This gives you an octal number equal in value to the binary number.

Binary fractions can also be converted to their octal equivalents using the same process, with one exception. The binary bits must be separated into groups of three beginning with the most significant bit. For example, the binary fraction 0.011101 is converted into its octal equivalent.

Again, you must first separate the binary number into

0.011101_2
REWRITE AS
MSB LSB
0.011 101
YIELDS
0.35_8

groups of three beginning at the radix (binary) point. Then covert each 3-bit group into its octal equivalent.

To separate binary number into 3-bit groups when the number does not contain the necessary bits, add zeros to the number until the number can be separated into 3-bit groups. For example, binary number 10010101.1011 is converted into its octal equivalent.

10010101.1011_2
REWRITE AS
MSB LSB
010 010 101.101 100
YIELDS
225.54_8

As before, the integer part of the number is separated into 3-bit groups, beginning at the radix (binary) point. Note that the third group contains only two bits. However, a zero can be added to the group without changing the value of the binary number. Next, the fractional part of the number is separated into 3-bit groups, beginning at the radix (binary) point. Note that the second group contains only one bit. By adding two zeros to the group, the group is complete with no change in the value of the binary number.

Note: Whenever you add zeros to a **binary integer,** *always place them to the* **left** *of the most significant bit. When you add zeros to a* **binary fraction,** *always place them to the* **right** *of the least significant bit.*

After you have formed the 3-bit groups, convert each group into its octal equivalent. This gives you an octal number equal in value to the binary number. Now convert binary number 1101110.01 into its octal equivalent.

1101110.01_2
REWRITE AS
MSB LSB
001 101 110.010
YIELDS
156.2_8

Separate the integer and fraction into 3-bit groups, adding zeros as necessary. Then convert each 3-bit group to octal. **Never** shift the radix (binary) point in order to form 3-bit groups.

Converting octal to binary is just the opposite of the previous process. You simply convert each octal number into its 3-bit binary equivalent. For example, convert the octal number 75.3 into its binary equivalent.

75.3_8
YIELDS
MSB LSB
111 101.011
REWRITE AS
111101.011_2

The above example is a simple conversion. Now a more complex octal number (1752.714) will be converted to a binary number.

1752.714_8
YIELDS
MSB LSB
001 111 101 010.111 001 100
REWRITES AS
1111101010.1110011_2

Again, each octal digit is converted into its 3-bit binary equivalent. However, in this example, there are two insignificant zeros in front of the MSB and after the LSB. Since these zeros have no value, they should be removed from the final result.

WHAT YOU HAVE LEARNED

1. A basic distinguishing feature of a number system is its *base* or *radix*. The base indicates the number of characters of digits used to represent quantities in that number system. A *subscript* may be used to show the base a number has been written in.
2. An *integer* is any of the natural, whole numbers and their negatives, or zero. *Decimal fractions* are numbers whose positions have weights that are negative powers of ten.
3. The *radix* separates the *integer* and fractional parts of a number. In our *decimal* system we refer to the radix as a "decimal point." In the *octal* system, for example, it might be called an "octal point."
4. A *binary* system contains two elements or states. In a number system it is expressed as a base of 2, using the digits 0 and 1. Binary digits are called *bits* and microprocessors operate on groups of *bits* which are referred to as words.
5. To *convert* a *binary* number *into* its *decimal* equivalent, add together the weights of the positions in the number where binary 1's occur.
6. To *convert* a *decimal* number *into* its *binary* equivalent, successively divide the number by the desired base. Note the *remainders*. The remainders will form the equivalent binary number.
7. The *octal* numbering system has a base (radix) of 8 and uses the digits 0 through 7. As with the binary system, each digit position of an octal number carries a positional weight which determines the magnitude of that number.
8. *Conversions from decimal to octal* are accomplished in the same manner as decimal to binary with one exception: The base number is now 8 rather than 2.
9. You may represent a 3-bit binary number with a 1-digit octal number. To *convert from binary to octal* first separate the number into groups containing three bits, beginning with the least significant bit. Then convert each 3-bit group into its octal equivalent. This will give you an octal number equal in value to the binary number.
10. In converting as above, to separate the binary number into 3-bit groups when the number does not contain the necessary bits, *add zeros* to the number until the number can be separated into 3-bit groups.

HEXADECIMAL NUMBER SYSTEM

Hexadecimal is another number system that is often used

with microprocessors. It is similar in value structure to the octal number system, and thus allows easy conversion with the binary number system. Because of this feature and the fact that hexadecimal simplifies data entry and display to a greater degree than octal, you will use hexadecimal more often than any other number system in this course. As the name implies, hexadecimal has a base (radix) of 16_{10}. It uses digits 0 through 9 and the letters A through F.

The letters are used because it is necessary to represent 16_{10} different values with a single digit for each value. Therefore, the letters A through F are used to represent the number values 10_{10} through 15_{10}. The following discussion will compare the decimal number system with the hexadecimal number system.

All of the numbers are of equal value between systems ($0_{10} = 0_{16}$, $3_{10} = 3_{16}$, $9_{10} = 9_{16}$, etc.). For numbers greater than 9, this relationship exists: $10_{10} = A_{16}$, $11_{10} = B_{16}$, $12_{10} = C_{16}$, $13_{10} = D_{16}$, $14_{10} = E_{16}$, and $15_{10} = F_{16}$. Using letters in counting may appear awkward until you become familiar with the system. Figure 1 illustrates the relationship between decimal and hexadecimal integers,

DECIMAL	HEXADECIMAL	BINARY
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010
19	13	10011
20	14	10100
21	15	10101
22	16	10110
23	17	10111
24	18	11000
25	19	11001
26	1A	11010
27	1B	11011
28	1C	11100
29	1D	11101
30	1E	11110
31	1F	11111
32	20	100000
33	21	100001
34	22	100010
35	23	100011

Figure 1

DECIMAL	HEXADECIMAL	BINARY
0.00390625	0.01	0.00000001
0.0078125	0.02	0.0000001
0.01171875	0.03	0.00000011
0.015625	0.04	0.000001
0.01953125	0.05	0.00000101
0.0234375	0.06	0.0000011
0.02734375	0.07	0.00000111
0.03125	0.08	0.00001
0.03515625	0.09	0.00001001
0.0390625	0.0A	0.0000101
0.04296875	0.0B	0.00001011
0.046875	0.0C	0.00011
0.05078125	0.0D	0.00001101
0.0546875	0.0E	0.0000111
0.05859375	0.0F	0.00001111
0.0625	0.1	0.0001
0.06640625	0.11	0.00010001
0.0703125	0.12	0.0001001
0.07421875	0.13	0.00010011
0.078125	0.14	0.000101
0.08203125	0.15	0.00010101
0.0859375	0.16	0.0001011
0.08984375	0.17	0.00010111
0.09375	0.18	0.00011
0.09765625	0.19	0.00011001
0.1015625	0.1A	0.0001101
0.10546875	0.1B	0.00011011
0.109375	0.1C	0.000111
0.11328125	0.1D	0.00011101
0.1171875	0.1E	0.0001111
0.12109375	0.1F	0.00011111
0.125	0.2	0.001

Figure 2

while Figure 2 illustrates the relationship between decimal and hexadecimal fractions.

As with the previous number systems, each digit position of a hexadecimal number carries a positional weight which determines the magnitude of that number. The weight of each position is determined by some power of the number system base (in this example, 16_{10}). The total quantity of a number can be evaluated by considering the specific digits and the weights of their positions. (Refer to Figure 3 for a condensed listing of powers of 16_{10}.) For example, the hexadecimal number E5D7.A3 can be written with positional notation as follows:

$$(E \times 16^3) + (5 \times 16^2) + (D \times 16^1) + (7 \times 16^0) + (A \times 16^{-1}) + (3 \times 16^{-2})$$

The decimal value of the hexadecimal number E5D7.A3 is determined by multiplying each digit by its positional weight and adding the results. As with the previous number systems, the radix (hexadecimal) point separates the integer from the fractional part of the number.

$(14 \times 4096) + (5 \times 256) + (13 \times 16) + (7 \times 1) + (10 \times 1/16) + (3 \times 1/256) = 57344 + 1280 + 208 + 7 + 0.625 + 0.01171875 = 58839.63671875_{10}$

Conversion From Decimal to Hexadecimal

Decimal to hexadecimal conversion is accomplished in

$16^{-4} = 1/65536 = 0.0000152587890625_{10}$

$16^{-3} = 1/4096 = 0.000244140625_{10}$

$16^{-2} = 1/256 = 0.00390625_{10}$

$16^{-1} = 1/16 = 0.0625_{10}$

$1_{10} = 16^0$
$16_{10} = 16^1$
$256_{10} = 16^2$
$4096_{10} = 16^3$
$65536_{10} = 16^4$
$1048576_{10} = 16^5$
$16777216_{10} = 16^6$

Figure 3

the same manner as decimal to binary or octal, but with a base number of 16_{10}. As an example, the decimal number 156 is converted into its hexadecimal equivalent.

$156 \div 16 = 9$ with remainder $12 = C \leftarrow$ LSD
$9 \div 16 = 0$ $9 = 0 \leftarrow$ MSD

Divide the decimal number by 16_{10} and note the remainder. If the remainder exceeds 9, convert the 2-digit number to its hexadecimal equivalent ($12_{10} = C$ in this example). Then divide the quotient by 16 and again note the remainder. Continue dividing until a quotient of 0 results. Then collect the remainders beginning with the last or most significant digit (MSD) and proceed to the first or least significant digit (LSD). The number $9C_{16} = 156_{10}$. NOTE: The letter H after a number is sometimes used to indicate hexadecimal. However, this course will always use the subscript 16.

To further illustrate this, the decimal number 47632 is converted into its hexadecimal equivalent.

$47632 \div 16 = 2977$ with remainder $0 = 0 \leftarrow$ LSD
$2977 \div 16 = 186$ $1 = 1$
$186 \div 16 = 11$ $10 = A$
$11 \div 16 = 0$ $11 = B \leftarrow$ MSD

The division process continues until a quotient of 0 results. The remainders are collected, producing the number $BA10_{16} = 47632_{10}$. Remember, any remainder that exceeds the digit 9 must be converted to its letter equivalent. (In this example, 10 = A, and 11 = B.)

To convert a decimal fraction to a hexadecimal fraction, multiply the fraction successively by 16_{10} (hexadecimal base). As an example, the decimal fraction 0.78125 is converted into its hexadecimal equivalent.

$0.78125 \times 16 = 12.5 = 0.5$ with overflow $12 = C \leftarrow$ MSD
$0.50000 \times 16 = 8.0 = 0$ $8 = 8 \leftarrow$ LSD

Multiply the decimal by 16_{10}. If the product exceeds one, subtract the integer (overflow) from the product. If the "overflow" exceeds 9, convert the 2-digit number to its hexadecimal equivalent. Then multiply the product fraction by 16_{10} and again note any overflow. Continue multiplying until an overflow, with 0 for a fraction, results. Remember, you can not always obtain 0 when you multiply by 16. Therefore, you should only continue the conversion to the accuracy or precision you desire. Collect the conversion overflows beginning at the radix point with the MSD and proceed to the LSD. The number $0.C8_{16} = 0.78125_{10}$.

Now the decimal fraction 0.136 will be converted into its hexadecimal equivalent with five-place precision.

$0.136 \times 16 = 2.176 = 0.176$ overflow $2 = 2 \rightarrow$ MSD
$0.176 \times 16 = 2.816 = 0.816$ $2 = 2$
$0.816 \times 16 = 13.056 = 0.056$ $13 = D$
$0.056 \times 16 = 0.896 = 0.896$ $0 = 0$
$0.896 \times 16 = 14.336 = 0.336$ $14 = E \rightarrow$ LSD

The number $0.22D0E_{16}$ approximately equals 0.136_{10}. If you convert $0.22D0E_{16}$ back to decimal (using positional notation), you will find $0.22D0E_{16} = 0.135999679565429-6875_{16}$. This example shows that extending the precision of your conversion is of little value unless extreme accuracy is required.

As shown in this section, conversion of an integer from decimal to hexadecimal requires a different technique than for conversion of a fraction. Therefore, when you convert a hexadecimal number composed of an integer and a fraction, you must separate the integer and fraction, then perform the appropriate operation on each. After you convert them, you must recombine the integer and fraction. For example, the decimal number 124.78125 is converted into its hexadecimal equivalent.

$124.78125_{10} = 124_{10} + 0.78125_{10}$
$124 \div 16 = 7$ with remainder $12 = C \leftarrow$ LSD
$7 \div 16 = 0$ $7 = 7 \leftarrow$ MSD

$\boxed{124_{10} = 7C_{16}}$

$0.78125 \times 16 = 12.5 = 0.5$ overflow $12 = C \leftarrow$ MSD
$0.50000 \times 16 = 8.0 = 0$ overflow $8 = 8 \leftarrow$ LSD

$\boxed{0.78125_{10} = 0.C8_{16}}$

$124.78125_{10} = 124_{10} + 0.78125_{10} =$
$7C_{16} + 0.C8_{16} = 7C.C8_{16}$

First separate the decimal integer and fraction. Then convert the integer and fraction to hexadecimal.

Finally, recombine the integer and fraction.

Converting Between the Hexadecimal and Binary Number Systems

Previously, the octal number system was described as an excellent shorthand form to express large binary quantities. This method is very useful with many microprocessors. The trainer used with this course uses the hexadecimal number system to represent binary quantities. As a result, frequent conversions from binary-to-hexadecimal are necessary. Figures 1 and 2 illustrate the relationship between hexadecimal and binary integers and fractions.

As you know, four bits of a binary number exactly equal 16_{10} value combinations. Therefore, you can represent a 4-bit binary number with a 1-digit hexadecimal number:

$1101_2 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) =$
$8 + 4 + 0 + 1 = 13_{10} = D_{16}$

Because of this relationship, converting binary to hexadecimal is simple and straightforward. For example, binary number 10110110 is converted into its hexadecimal equiva-

lent.

10110110_2
REWRITE AS
MSB LSB
1011 0110
YIELDS
$B6_{16}$

To convert a binary number to hexadecimal, first separate the number into groups containing four bits, beginning with the least significant bit. Then convert each 4-bit group into its hexadecimal equivalent. Don't forget to use letter digits as required. This gives you a hexadecimal number equal in value to the binary number.

Now convert a larger binary number (10101101101) into its hexadecimal equivalent.

10110110_2
REWRITE AS
MSB LSB
0101 0110 1101
YIELDS
$56D_{16}$

Again, the binary number is separated into 4-bit groups beginning with the LSB. However, the third group contains only three bits. Since each group must contain four bits, a zero must be added after the MSB. The third group will then have four bits with no change in the value of the binary number. Now each 4-bit group can be converted into its hexadecimal equivalent. **Whenever you add zeros to a binary integer, always place them to the left of the most significant bit.**

Binary fractions can also be converted to their hexadecimal equivalents using the same process, with one exception; the binary bits are separated into groups of four, beginning with the most significant bit (at the radix point). For example, the binary fraction 0.01011011 is converted into its hexadecimal equivalent.

0.01011011_2
REWRITE AS
MSB LSB
0.0101 1011
YIELDS
$0.5B_{16}$

Again, you must separate the binary number into groups of four, beginning with the radix point. Then convert each 4-bit group into its hexadecimal equivalent. This gives you a hexadecimal number equal in value to the binary number.

Now convert a larger binary fraction (0.1101001101) into its hexadecimal equivalent.

0.1101001101_2
REWRITE AS
MSB LSB
0.1101 0011 0100
YIELDS
$0.D34_{16}$

Separate the binary number into 4-bit groups, beginning at the radix (binary) point (MSB). Note that the third group contains only two bits. Since each group must contain four bits, two zeros must be added after the LSB. The third group will then have four bits with no change in the value of the binary number. Now, each 4-bit group can be converted into its hexadecimal equivalent. **Whenever you add zeros to a binary fraction, always place them to the right of the least significant bit.**

Now, a binary number containing both an integer and a fraction (110110101.01110111) will be converted into its hexadecimal equivalent.

110110101.01110111_2
REWRITE AS
MSB LSB
0001 1011 0101.0111 0111
YIELDS
$1B5.77_{16}$

The integer part of the number is separated into groups of four, **beginning** at the radix point. Note that three zeros were added to the third group to complete the group. The fractional part of the number is separated into groups of four, **beginning** at the radix point. (No zeros were needed to complete the fractional groups.) The integer and fractional 4-bit groups are then converted to hexadecimal. The number $110110101.01110111_2 = 1B5.77_{16}$. **Never** shift the radix point in order to form 4-bit groups.

Converting hexadecimal to binary is just the opposite of the previous process; simply convert each hexadecimal number into its 4-bit binary equivalent. For example, convert the hexadecimal number 8F.41 into its binary equivalent.

$8F.41_{16}$
YIELDS
MSB LSB
1000 1111.0100 0001
REWRITE AS
10001111.01000001_2

Convert each hexadecimal digit into a 4-bit binary number. Then condense the 4-bit groups to form the binary value equal to the hexadecimal value. The number $8F.41_{16} = 10001111.01000001_2$.

Now, the hexadecimal number 175.4E will be converted into its binary equivalent.

$175.4E_{16}$
YIELDS
MSB LSB
0001 0111 0101.0100 1110
REWRITE AS
101110101.0100111_2

Again, each hexadecimal digit is converted into its 4-bit binary equivalent. However, in this example there are three insignificant zeros in front of the MSB and one after the LSB. Since these zeroes have no value, they should be removed from the final result.

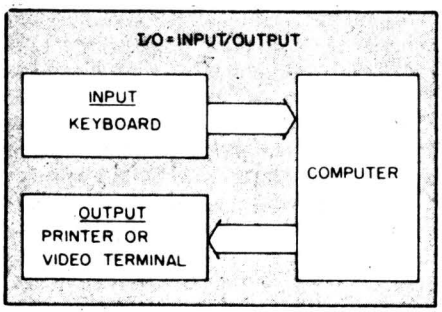
BINARY CODES

Converting a decimal number into its binary equivalent is

One method of information entry is by a keyboard. MicroAge is one of many makers.

balanced with an integer basic.

The first of the really useful BASICs came out of SWTP Co. and was known as 4K BASIC. It was pure dynamite compared to the integer BASICs. SWTP Co. was to upgrade it to an 8K BASIC, meaning it utilized about 6.5K of memory for the language, leaving some 1.5K of memory for storage and operations. The SWTP Co. 8K BASIC was as good as most BASICs in school computers, and better than many. It pointed the way to go for all hobby BASICs.



Getting data in and out of the computer is the job of the I/O (Input/Output) terminals. The data is entered on a typewriter-like keyboard and the letters and numbers are translated into electronic signals that the computer can understand. The results of the data processing are printed on paper by a printer or displayed on a video terminal. They also keep a listing of the data and instructions.

Heathkit was later to develop an outstanding extended BASIC for their H8 hobbyist computer which outclassed most of the BASICs used in schools and many time-share (rental time) systems.

Between SWTP Co. and Heathkit, BASIC was virtually welded in place as the standard hobby computer language. Aside from its rather extensive capabilities, and aside from the fact it is easily upgraded (which is contrary to what many experts claimed), BASIC requires relatively modest memory, and memory is the most expensive part of a computer system.

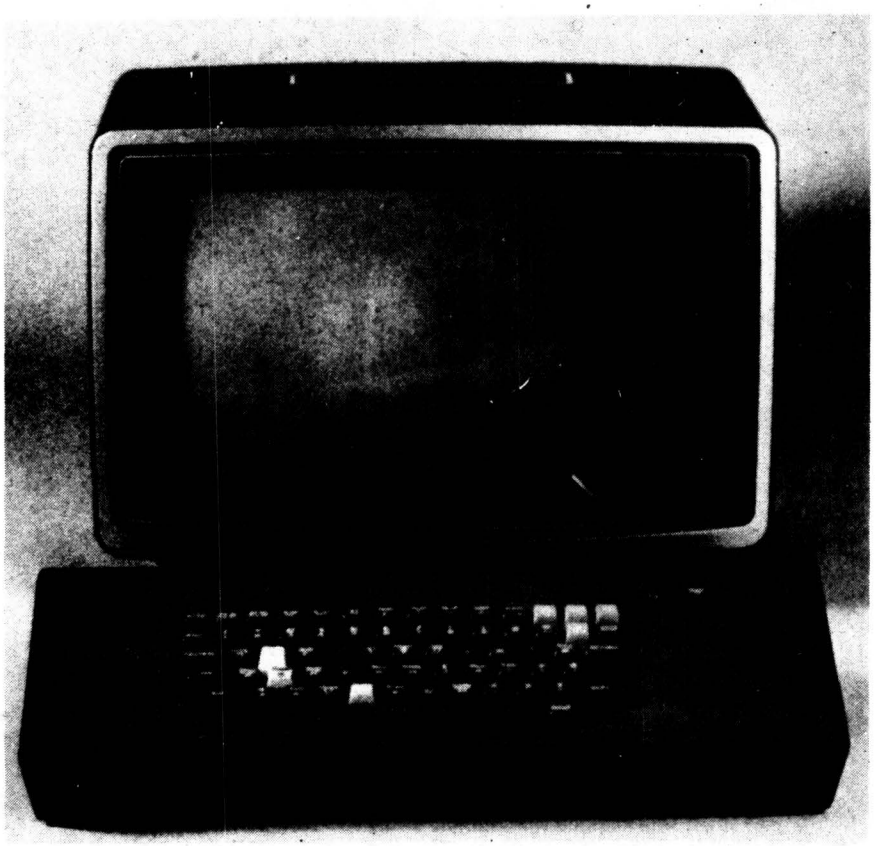
Memories. As the costs of memory and massive data handling systems, such as the disc, are reduced, we will start to see the appearance of other high level languages such as Fortran in hobby computers. But until both memory and the language itself represents just pocket money, BASIC will remain the hobby computer language.

So keep in mind, if you plan any serious work with hobby computers, you need not only the hardware, but an excellent floating point (non-integer) BASIC specifically written for your computer. (You generally cannot run a BASIC written on one computer on another utilizing a different CPU or addressing system.)

Keep in mind as you consider which hobby computer equipment to use that BASIC can either be programmed into the computer via a punched tape (teletype tape) or audio cassette, or it can be a permanent part of the computer itself, built into pre-programmed devices called ROMs—Read Only Memory. Computers such as the Radio Shack TRS-80, Apple II, and Ohio Scientific (OSI) Challenger have the BASIC in ROM, available for use the instant power is turned on. The one disadvantage to ROM BASIC is that it isn't easily modified or upgraded. Either you get a new ROM, or upgrade with software through a cassette tape. Programmable BASIC, such as used by SWTP Co and Heathkit, are upgraded by simply purchasing the new version as a low cost audio cassette. Later in this issue we will illustrate the various means by which software is recorded and played into the computer.

Hardware. Now that we're familiar with a basic computer and BASIC (no pun intended), let's move back into hardware. All a computer can do is sit quietly with its power light glowing until there's some means whereby the user can communicate with the computer. Communication means a terminal, consisting of a keyboard and some means to display the computer's output. The display can be a television set modified for computer use through an inexpensive device called a "modulator" that connects directly to the antenna terminals, the display can be a television monitor, which we term a CRT (cathode ray tube), or it can be a printer that provides a "hard copy" printed on a piece of paper.

Though the CRT terminal is probably the lowest in cost, and extremely fast, it doesn't provide a permanent record unless you take a photograph of the CRT, or use a device that makes



For those who don't have the time or desire to assemble a system, a packaged, personal computer such as the Compucolor II may be the answer.

called "coding." A decimal number is expressed as a binary code or binary number. The **binary number system,** as discussed, is known as the pure binary code. This name distinguishes it from other types of binary codes. This section will discuss some of the other types of binary codes used in computers.

Binary Coded Decimal

The decimal number system is easy to use because it is so familiar. The binary number system is less convenient to use because it is less familiar. It is difficult to quickly glance at a binary number and recognize its decimal equivalent. For example, the binary number 1010011 represents the decimal number 83. It is difficult to tell immediately by looking at the number what its decimal value is. However, within a few minutes, using the procedures described earlier, you could readily calculate its decimal value. The amount of time it takes to convert or recognize a binary number quantity is a distinct disadvantage in working with this code despite the numerous hardware advantages. Engineers recognized this problem early and developed a special form of binary code that was more compatible with the decimal system. Because so many digital devices, instruments and equipment use decimal input and output, this special code has become very widely used and accepted. This special compromise code is known as binary coded decimal (BCD). The BCD code combines some of the characteristics of both the binary and decimal number systems.

8421 BCD Code. The BCD code is a system of representing the decimal digits 0 through 9 with a four-bit binary code. This BCD code uses the standard 8421 position **weighting system** of the pure binary code. The standard 8421 BCD code and the decimal equivalents are shown in Figure 4 along with a special Gray code that will be described later. As with the pure binary code, you can convert the BCD numbers into their decimal equivalents by simply adding together the weights of the bit positions whereby the binary 1's occur. Note, however, that there are only ten possible valid 4-bit code arrangements. The 4-bit binary numbers representing the decimal numbers 10 through 15 are invalid in the BCD system.

DECIMAL	8421 BCD	GRAY	BINARY
0	0000	0000	0000
1	0001	0001	0001
2	0010	0011	0010
3	0011	0010	0011
4	0100	0110	0100
5	0101	0111	0101
6	0110	0101	0110
7	0111	0100	0111
8	1000	1100	1000
9	1001	1101	1001
10	0001 0000	1111	1010
11	0001 0001	1110	1011
12	0001 0010	1010	1100
13	0001 0011	1011	1101
14	0001 0100	1001	1110
15	0001 0101	1000	1111

Figure 4

To represent a decimal number in BCD notation, substitute the appropriate 4-bit code for each decimal digit. For example, the decimal integer 834 in BCD would be 1000 0011 0100. Each decimal digit is represented by its equivalent 8421 4-bit code. A space is left between each 4-bit group to avoid confusing the BCD format with the pure binary code. This method of representation also applies to decimal fractions. For example, the decimal fraction 0.764 would be 0.0111 0110 0100 in BCD. Again, each decimal digit is represented by its equivalent 8421 4-bit code, with a space between each group.

An advantage of the BCD code is that the ten BCD code combinations are easy to remember. Once you begin to work with binary numbers regularly, the BCD numbers may come to you as quickly and automatically as decimal numbers. For that reason, by simply glancing at the BCD representation of a decimal number you can make the conversion almost as quickly as if it were already in decimal form. As an example, convert a BCD number into its decimal equivalent.

$$0110\ 0010\ 1000.1001\ 0101\ 0100 = 628.954_{10}$$

The BCD code simplifies the man-machine interface but it is less efficient than the pure binary code. It takes more bits to represent a given decimal number in BCD than it does with pure binary notation. For example, the decimal number 83 in pure binary form is 1010011. In BCD code the decimal number 83 is written as 1000 0011.. In the pure binary code, it takes only seven bits to represent the number 83. In BCD form, it takes eight bits. It is inefficient because, for each bit in a data word, there is usually some digital circuitry associated with it. The extra circuitry associated with the BCD code costs more, increases equipment complexity, and consumes more power. Arithmetic operations with BCD numbers are also more time consuming and complex than those with pure binary numbers. With four bits of binary information, you can represent a total of $2^4 = 16$ different states or the decimal number equivalents 0 through 15. In the BCD system, six of these states (10-15), are wasted. When the BCD number system is used, some efficiency is traded for the improved communications between the digital equipment and the human operator.

Decimal-to-BCD conversion is simple and straightforward. However, binary-to-BCD conversion is not direct. An intermediate conversion to decimal must be performed first. For example, the binary number 1011.01 is converted into its BCD equivalent.

First the binary number is converted to decimal.

$$1011.01_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2})$$
$$= 8 + 0 + 2 + 1 + 0 + 0.25$$
$$= 11.25_{10}$$

Then the decimal result is converted to BCD.

$$11.25_{10} = 0001\ 0001.0010\ 0101$$

To convert from BCD to binary, the previous operation is reversed. For example, the BCD number 1001 0110.0110 0010 0101 is converted into its binary equivalent.

First, the BCD number is converted to decimal. 1001

$0110.0110\ 0010\ 0101 = 96.625_{10}$. Then the decimal result is converted to binary. $96.625_{10} = 96_{10} + 0.625_{10}$

$96 \div 2 = 48$	with remainder 0	← LSB
$48 \div 2 = 24$	0	
$24 \div 2 = 12$	0	
$12 \div 2 = 6$	0	
$6 \div 2 = 3$	0	
$3 \div 2 = 1$	1	
$1 \div 2 = 0$	1	← MSB

$\boxed{96_{10} = 1100000_2}$

$0.625 \times 2 = 1.25 = 0.25$	with overflow 1	← MSB
$0.250 \times 2 = 0.50 = 0.50$	0	
$0.500 \times 2 = 1.00 = 0$	1	← LSB

$\boxed{0.625_{10} = 0.101_2}$

$96.625_{10} = 96_{10} + 0.625_{10} = 1100000_2 + 0.101_2 = 1100000.101_2$
Therefore:
$1001\ 0110.0110\ 0010\ 0101 = 96.625_{10} = 1100000.101_2$

Because the intermediate decimal number contains both an integer and fraction, each number portion is converted as described under "Binary Number System." The binary sum (integer plus fraction) 1100000.101 is equivalent to the BCD number 1001 0110.0110 0010 0101.

Special Binary Codes

Besides the standard pure binary coded form, the BCD numbering system is by far the most widely-used digital code. You will find one or the other in most of the applications that you encounter. However, there are several other codes that are used for special applications, such as the "Gray Code."

The Gray Code is a widely-used, non-weighted code system. Also known as the cyclic, unit distance or reflective code, the Gray code can exist in either the pure binary or BCD formats. The Gray code is shown in Figure 4. As with the pure binary code, the first ten codes are used in BCD operations. Notice that there is a change in only one bit from one code number to the next in sequence. You can get a better idea about the Gray code sequence by comparing it to the standard 4-bit 8421 BCD code and the pure binary code also shown in Figure 4. For example, consider the change from 7 (0111) to 8 (1000) in the pure binary code. When this change takes place, all bits change. Bits that were 1's are changed to 0's and 0's are changed to 1's. Now notice the code change from 7 to 8 in the Gray code. Here 7 (0100) changes to 8 (1100). Only the first bit changes.

The Gray code is generally known as an error minimizing code because it greatly reduces confusion in the electronic circuitry when changing from one state to the next. When binary codes are implemented with electronic circuitry, it takes a finite period of time for bits to change from 0 to 1 or 1 to 0. These state changes can create timing and speed problems. This is particularly true in the standard 8421 codes where many bits change from one combination to the next. When the Gray code is used, however, the timing and speed errors are greatly minimized because only one bit changes at a time. This permits code circuitry to operate at higher speeds with fewer errors.

The biggest disadvantage of the Gray code is that it is difficult to use in arithmetic computations. Where numbers must be added, subtracted or used in other computations, the Gray code is not applicable. In order to perform arithmetic operations, the Gray code number must generally be converted into pure binary form.

Alphanumeric Codes

Several binary codes are called alphanumeric codes because they are used to represent characters as well as numbers. The two most common codes that will be discussed are ASCII and BAUDOT.

ASCII Code. The American Standard Code for Information Interchange commonly referred to as ASCII, is a special form of binary code that is widely used in microprocessors and data communications equipment. A new name for this code that is becoming more popular is the American National Standard Code for Information Interchange (ANSCII). However, this course will use the most recognized term, ASCII. ASCII is a 6-bit binary code that is used in transferring data between microprocessors and their peripheral devices, and in communicating data by radio and telephone. With six bits, a total of $2^6 = 64$ different characters can be represented. These characters comprise decimal numbers 0 through 9, upper-case letters of the alphabet, plus other special characters used for punctuation and data control. A 7-bit code called full ASCII, extended ASCII, or USASCII can be represented by $2^7 = 128$ different characters. In addition to the characters and numbers genetared by 6-bit ASCII, 7-bit ASCII contains lower-case letters of the alphabet, and additional characters for punc-

NOTES:

(1) Depending on the machine using this code, the symbol may be a circumflex, an up-arrow, or a horizontal parenthetical mark.
(2) Depending on the machine using this code, the symbol may be an underline, a back-arrow, or a heart.
(3) Explanation of special control functions in colums 0, 1, 2, and 7.

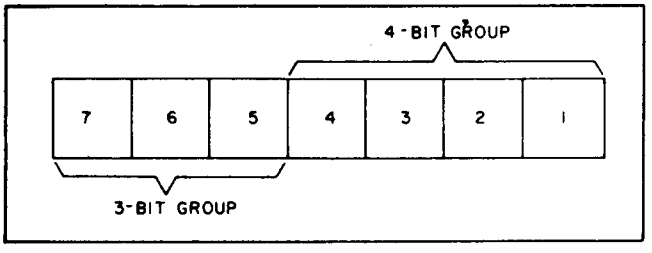
NUL	Null	DLE	Data Link Escape
SOH	Start of Heading	DC1	Device Control 1
STX	Start of Text	DC2	Device Control 2
ETX	End of Text	DC3	Device Control 3
EOT	End of Transmission	DC4	Device Control 4
ENQ	Enquiry	NAK	Negative Acknowledge
ACK	Acknowledge		
BEL	Bell (audible signal)	SYN	Synchronous Idle
BS	Backspace	ETB	End of Transmission
HT	Horizontal Tabulation (punched card skip)	CAN	Block Cancel
LF	Line Feed	EM	End of Medium
VT	Vertical Tabulation	SUB	Substitute
FF	Form Feed	ESC	Escape.
CR	Carriage Return	FS	File Separator
SO	Shift Out	GS	Group Separator
SI	Shift In	RS	Record Separator
SP	Space (blank)	US	Unit Separator
		DEL	Delete

tuation and control. The 7-bit ASCII code is shown in Figure 5.

ROW	COLUMN / BITS 4321 / 765	0(3) 000	1(3) 001	2(3) 010	3 001	4 100	5 101	6 110	7(3) 111
0	0000	NUL	DLE	SP	0	@	P	/	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	v
10	1010	LF	SUB	*	:	J	Z	j	z
11	1011	VT	ESC	+	;	K	[	k	{
12	1100	FF	FS	,	<	L	/	l	|
13	1101	CR	GS	-	=	M	]	m	}
14	1110	SO	RS	.	>	N	∪(1)	n	~
15	1111	SI	US	/	?	O	←(2)	o	DEL

Figure 5

The 7-bit ASCII code for each number, letter or control function is made up of a 4-bit group and a 3-bit group. Figure 6 shows the arrangement of these two groups and the numbering sequence. The 4-bit group is on the right and bit 1 is the LSB. Note how these groups are arranged in rows and columns in Figure 5.



To determine the ASCII code for a given number letter or control operation, locate that item in the table. Then use the 3- and 4-bit codes associated with the row and column in which the item is located. For example, the ASCII code for the letter L is 1001100. It is located in column 4, row 12. The most significant 3-bit group is 100, while the least significant 4-bit group is 1100. When 6-bit ASCII is used, the 3-bit group is reduced to a 2-bit group as shown in Figure 7.

In 7-bit ASCII code, an eighth bit is often used as a **parity** or check bit to determine if the data (character) has been transmitted correctly. The value of this bit is determined by the type of parity desired. **Even parity** means resent two separate characters. As shown in Figure 1-15, one set of 5-bit codes represents the 26 upper-case alphabet letters. The same 5-bit codes also represent various figures and the decimal number series 0 through 9.

The remaining six 5-bit codes are used for machine control and do not have a secondary function. Two of these 5-bit codes determine which of the 26 double (letter/figure) characters can be transmitted/received. Bit number 11111 forces the printer to recognize all following 5-bit codes as **letters.** Bit number 11011 forces **figure** recognition of all

NOTES:

(1)Depending on the machine using this code, the symbol may be a circumflex, an up-arrow, or a horizontal parenthetical mark.
(2)Depending on the machine using this code, the symbol may be an underline, a back-arrow, or a heart.
(3)SP—Space (blank) for machine control.

ROW	COLUMN / BITS 4321 / 65	0 10	1 11	2 00	3 01
0	0000	SP(3)	0	@	P
1	0001	!	1	A	Q
2	0010	"	2	B	R
3	0011	#	3	C	S
4	0100	$	4	D	T
5	0101	%	5	E	U
6	0110	&	6	F	V
7	0111	'	7	G	W
8	1000	(	8	H	X
9	1001	)	9	I	Y
10	1010	*	:	J	Z
11	1011	+	;	K	[
12	1100	,	<	L	/
13	1101	-	=	M	]
14	1110	.	>	N	∪(1)
15	1111	/	?	O	←(2)

Figure 7

the following 5-bit codes. For example, to type 56 NORTH 10 STREET, the following method is used.

BAUDOT Code. While the ASCII code is used almost exclusively with microprocessor peripheral devices (CRT display, keyboard terminal, paper punch/reader, etc.), there are many older printer peripherals that use the 5-bit BAUDOT code. With five data bits, this code can represent only $2^5 = 32$ different characters. To obtain a greater

POSITIVE POWERS OF 16

n	16^n
0	1
1	16
2	256
3	409 6
4	655 36
5	104 857 6
6	167 772 16
7	268 435 456
8	429 496 729 6

NEGATIVE POWERS OF 16

n	16^{-n}
0	1.0
1	0.062 5
2	0.003 906 25
3	0.000 244 140 625
4	0.000 015 258 789 062 5

Bit Numbers	Letters Case	Figures Case
5 4 3 2 1	Blank	Blank
0 0 0 0 0	E	3
0 0 0 0 1	Line Feed	Line Feed
0 0 0 1 0	A	–
0 0 0 1 1	Space	Space
0 0 1 0 0	S	Bell
0 0 1 0 1	I	8
0 0 1 1 0	U	7
0 0 1 1 1	Car. Ret.	Car. Ret.
0 1 0 0 0	D	$
0 1 0 0 1	R	4
0 1 0 1 0	J	(Apos)'
0 1 0 1 1	N	(Comma),
0 1 1 0 0	F	!
0 1 1 0 1	C	:
0 1 1 1 0	K	(
0 1 1 1 1	T	5
1 0 0 0 0	Z	"
1 0 0 0 1	L	)
1 0 0 1 0	W	2
1 0 0 1 1	H	Stop
1 0 1 0 0	Y	6
1 0 1 0 1	P	0
1 0 1 1 0	Q	1
1 0 1 1 1	O	9
1 1 0 0 0	B	?
1 1 0 0 1	G	&
1 1 0 1 0	Figures	Figures
1 1 0 1 1	M	.
1 1 1 0 0	X	/
1 1 1 0 1	V	;
1 1 1 1 0	Letters	Letters
1 1 1 1 1		

Figure 8

character capability, 26 of the 5-bit codes are used to rep-

the sum of all the 1 bits, including the parity bit, is an even number. For example, if G is the character transmitted, the ASCII code is 1000111. Since four 1's are in the code, the parity bit is 0. The 8-bit code would be written 01000111.

POSITIVE POWERS OF 2

n	2^n	
0	1	
1	2	
2	4	
3	8	
4	16	
5	32	
6	64	
7	128	
8	256	
9	512	
10	1024	
11	2048	
12	4096	
13	8192	
14	16384	
15	32768	
16	65536	
17	13107	2
18	26214	4
19	52428	8
20	10485	76
21	20971	52
22	41943	04
23	83886	08
24	16777	216
25	33554	432
26	67108	864
27	13421	7728
28	26843	5456
29	53687	0912
30	10737	41824
31	21474	83648
32	42949	67296

decimal A.

3. Conversion from *decimal to hexadecimal* is done the same way as decimal to binary or octal. The decimal number is divided by decimal 16, and the remainder noted. The quotient is divided by 16 again, and again the remained is noted. Division by decimal 16 continues until the quotient is 0. The remainders, from least to most significant digit, are finally collected and written as the hexadecimal number.
4. Conversion from decimal *fractions to* hexadecimal is done in the same manner as decimal fractions to octal or binary. In this case, the fraction is successively multiplied by 16, and the overflow noted.
5. Four *bits* of a binary number exactly equal 16. So, a 4-bit binary number can be represented with a 1-digit hexadecimal number. Therefore, to *convert from binary to hexadecimal*: Begin with the *least* significant bit and separate the binary number into 4-bit groups and then convert each 4-bit group directly to its hexadecimal equivalent. This will give you the hexadecimal number equal in value to the binary.
6. Convert binary *fractions* to hexadecimal the same way, but instead of beginning with the least significant bit, the *most* significant bit should be the starting point.

POSITIVE POWERS OF 8

n	8^n
0	1
1	8
2	64
3	512
4	409 6
5	327 68
6	262 144
7	209 715 2
8	167 772 16

7. When you convert a decimal number to a binary equivalent, the process is referred to as *coding*. The *pure* binary code is one of many binary codes.
8. Binary coded decimals are easy to use. To represent a decimal number as a binary coded digit (*BCD*) the appropriate 4-bit code of pure binary numbering is substituted for each of the decimal digits. Advantages to this system is that it is a good *compromise* between men and machines, it is easy to learn and simpler than pure binary. However, it is much *less efficient* than pure binary. Circuits using it must be more complicated, less efficient, and more prone to time delays.
9. The *Gray Code* is a widely-used system. Only one bit changes from one number to the next one in sequence. It minimizes errors in electronic circuitry when it changes from one state to the next. However, it is quite difficult to use in arithmetic computations.
10. ASCII is the microcomputer, data-processing, *alphanumeric code*. Basically a binary system, it can represent alphanumeric characters through a variety of binary systems using six, seven, or eight bits.
11. *Parity* is the *check bit*, the last bit in a seven or eight-bit ASCII code. It is there to determine if the data has been sent correctly. There is *even* and *odd* parity but both accomplish the same end.
12. BAUDOT code is the ancestor of ASCII. It is hardly in use anymore. It is a 5-bit code and so can only represent 32 characters. Whether a given code represents a letter or a figure must be inputed at the keyboard of the terminal.

NEGATIVE POWERS OF 2

n	2^{-n}
0	1.0
1	0.5
2	0.25
3	0.125
4	0.0625
5	0.03125
6	0.01562 5
7	0.00781 25
8	0.00390 625
9	0.00195 3125
10	0.00097 65625
11	0.00048 82812 5
12	0.00024 41406 25
13	0.00012 20703 125
14	0.00006 10351 5625
15	0.00003 05175 78125
16	0.00001 52587 89062 5
17	0.00000 76293 94531 25
18	0.00000 38146 97265 625
19	0.00000 19073 48632 8125
20	0.00000 09536 74316 40625
21	0.00000 04768 37158 20312 5
22	0.00000 02384 18579 10156 25
23	0.00000 01192 09289 55078 125
24	0.00000 00596 04644 77539 0625
25	0.00000 00298 02322 38769 53125
26	0.00000 00149 01161 19384 76562 5
27	0.00000 00074 50580 59692 38281 25
28	0.00000 00037 25290 29846 19140 625
29	0.00000 00018 62645 14923 09570 3125
30	0.00000 00009 31322 57461 54785 15625
31	0.00000 00004 65661 28730 77392 57812 5
32	0.00000 00002 32830 64365 38696 28906 25

Programming with BASIC

BASIC is the most popular hobby computer language and we're going to make it easy for you to learn. This short course will have you planning and understanding simple computer programs, and leave you with the foundation necessary to set out on your own after greater complexity. The world of programming awaits—speak BASIC and enter!

You will start off slowly and simply, to kind of ease your way into things. There are several devices that are commonly used for communication between a computer and computer user.

Q1 What characteristics do these devices have in common?
(a) television screen
(b) a typewriter-like keyboard
(c) a steering wheel

A1 (b) a typewriter-like keyboard

A computer *terminal* provides the means for communicating with the computer. By means of a teletype or other terminal, a computer program and data may be communicated to a computer. When the program is *run* or processed, the computer sends signals to the terminal which provides *output*–that is, the results of processing the program. (Therefore, a terminal provides two-way communication between the computer and the user.)

Q2 A terminal provides ____-way communication between the computer and the user.

A2 two

The *teletypewriter* is the most common device used for communication between the computer and its user, and it is the most common *computer terminal*. The teletype is used much as an electric typewriter. It prints the numerals 1, 2, 3, 4, 5, 6, 7, 8, 9, and 0; the letters of the alphabet; and some special symbols. Letters are printed in upper case (capitals) only. You may not use the lower case L to stand for the numeral one.

Q3 To print numeral one, you depress
(a) the key marked 1
(b) lower case letter L

A3 (a) the key marked 1

So you want to know what computer programming is all about? Here's a computer program that will calculate a student's grade point average.


```
100 REMARK PRØGRAM TO CØMPUTE GRADE PØINT AVERAGE
110 PRINT "HØW MANY UNITS ØF A";
120 INPUT A
130 PRINT "HØW MANY UNITS ØF B";
140 INPUT B
150 PRINT "HØW MANY UNITS ØF C";
160 INPUT C
170 PRINT "HØW MANY UNITS ØF D";
180 INPUT D
190 PRINT "HØW MANY UNITS ØF F";
200 INPUT F
210 LET U=A+B+C+D+F
220 LET G=(4*A+3*B+2*C+1*D)/U
230 PRINT
240 PRINT "YØUR GRADE PØINT AVERAGE IS";G
999 END

RUN

HØW MANY UNITS ØF A?4
HØW MANY UNITS ØF B?6
HØW MANY UNITS ØF C?6
HØW MANY UNITS ØF D?0
HØW MANY UNITS ØF F?0

YØUR GRADE PØINT AVERAGE IS 2.875
```

This is the program.

This is the output or result of running the program above.

The program consists of 16 *statements*, each one on a separate line numbered 100–999. Each line begins with a line number. Following each line number is a statement that contains instruction to the computer.

This progam was typed a line at a time on the teletype (or other terminal) and was saved in the computer's memory. Then we told the computer to RUN the program; that is, to follow the instructions in the program. During the run the computer, following the instructions in the program, asked for information (called input) to be supplied by the computer user–how many units of A's, B's, C's, etc., were received? The program then directed the computer to do the computation and print the result. By the end of Chapter Two, you will be able to understand and use all the BASIC notation used in this program and more, so read on.

Q4 The distinct lines in a computer *program* are called _____.

A4 statements

The computer stores a program in its "memory." Before the computer user attempts to enter a new program into the computer, he will want to remove any previous instructions

that may be currently in the memory. To erase previous instructions in the computer, type the letters SCR, then press the key marked RETURN. SCR stands for SCRatch, and scratches out or erases (removes) any previous program in the computer.

Q5 Before a new program is typed into the computer, any old instructions held in the computer memory should be erased. To erase an old program, type SCR and press the key marked _____.

A5 RETURN

NOTE: Although most of the words and symbols in BASIC are the same for all computer systems that use BASIC, there are some exceptions. This is because there has not been a completely standardized form of the language that is used by all computer manufacturers. Common variations will be noted throughout this text. However, the concepts involved are the same, even though a particular code word or symbol may be different from that used here. When you have a grasp of BASIC, you will find it easy to make the substitutions necessary to use the particular computer system at hand, and a quick review of the BASIC reference manual for your system will provide you with any variations you need to know. For example, words such as NEW or CLEAR or START are used in place of SCRatch on some computer systems, but fortunately such variety is the exception rather than the rule.

Sample program:

```
1      LET A=5
10     LET B=10
135    LET C=A+B
277    PRINT A,B
852    PRINT C
9999   END
```

THIS PROGRAM, WRITTEN IN BASIC, CONSISTS OF SIX STATEMENTS. NOTE THAT EACH STATEMENT BEGINS WITH A *LINE NUMBER*.

From this example you can see that line numbers may range from 1 to 9999.

NOTE: The upper limit for line numbers is different on some computers.

The line numbers indicate to the computer the order in which it is to follow the instructions in the program. It is not necessary for line numbers to follow each other successively (e.g., 1, 2, 3, 4, . . .) as you can see by looking at the line numbers in the program in the previous frame. However, it is more common to number by ten's as we have in the program below. Then, if we wish, more statements may be easily inserted in the program between existing statements.

```
10 LET A=5
20 LET B=10
30 LET C=A+B
40 PRINT A,B
50 PRINT  C
99 END
```

THIS IS A COMMON WAY OF NUMBERING A PROGRAM. NOTE THAT THE LINE NUMBER FOR THE END STATEMENT IS 99. FOR CONVENIENCE, WE WILL USE 99 OR 999 OR 9999 FOR THE END STATEMENTS, DEPENDING ON THE SIZE OF THE OTHER LINE NUMBERS IN THE PROGRAM.

Nine new statements could be added between Lines 20 and 30; they would be consecutively numbered Line 21, 22, 23, 24, 25, 26, 27, 28, and 29.

```
10 PRINT 12 + 33
99 END
```

This is a very short program that is composed of only two statements. Each statement begins with a line number.

```
10 PRINT 12 + 33
99 END
```

In this mini-program, Line 10 instructs the computer to evaluate the numerical expression 12 + 33, (i.e., do the arithmetic) and to PRINT the result. When this program is run on the computer it will print the sum of 12 and 33.

The computer follows instructions in *line number order.* In the preceeding program, Line 10 is done before Line 99.

```
10 PRINT 12 + 33
99 END
```

If you are seated at the computer terminal, and have erased any previous programs in the computer, you are ready to type in this program. To enter the program, type the first line, then press the RETURN key. Then type the second line and press the RETURN key.

```
10 PRINT 12 + 33
99 END
```

Assume you have typed this program into the computer. Now you wish the computer to process the program. Type RUN and press the RETURN key. If you have not made any typing errors in entering the program, the computer will evaluate 12 + 33, print the result 45, and stop. Here is what you would see on the teletype printout.

```
10 PRINT 12 + 33
99 END
RUN
45
```

THE PROGRAM YOU TYPED IN.

Q6 What command, in any of the previous programs, tells the computer to begin to follow the instructions contained in the program?

A6 RUN

The program is not altered or erased from the computer's memory when you RUN it. Every time you type RUN and press RETURN the computer will RUN the program. Every time you RUN a program with the same information you will get the same result.

NOTE: There are some exceptions to the above rule.

Q7 As a rule, every time you RUN a program with the same information you will get _____ result.

A7 the same

```
10 PRINT "12 + 33"
99 END
```

Look at this program. Do you see the difference from the ones we have already studied? That's right—the numerical expression is enclosed in quotation marks.

Note that the computer evaluated these two programs

quite differently when it was told to RUN them.

```
10 PRINT "12 + 33"
99 END
RUN

12 + 33


10 PRINT 12 + 33
99 END
RUN

45
```

The statement PRINT 12 + 33 without quote marks tells the computer to evaluate the *numerical expression* 12 + 33 (i.e., do the arithmetic) and print the result as a decimal numeral.

The statement PRINT "12 + 33" with quote marks tells the computer to print the *string* enclosed in quotation marks *exactly as it appears.* No arthimetic is performed.

Q8 In BASIC, a string is information in a PRINT statement that is enclosed by _____ _____.

A8 quotation marks.

Q9 Fill in the blank as the computer would print it.

```
20 PRINT "186 - 58"
25 END
RUN
```

A9 186–58 (Note that the computer does not print the quotation marks.)

```
PRINT "MY HUMAN
UNDERSTANDS ME"
```

The underlined portion of the statement is a *string.* It is enclosed in quotation marks.

```
PRINT "12 + 33"
THIS IS A STRING. IT IS ENCLOSED IN QUOTATION
MARKS.

PRINT 12 + 33
THIS IS NOT A STRING. IT IS A NUMERICAL EXPRESSION.
```

A *string* may include

(a) numerals (0, 1, 2, . . .)
(b) letters (A, B, C, . . .)
(c) special characters (+, –, *, /, ↑, comma, period, semicolon, etc.)

Since quotation marks define the beginning and end of a string, they cannot be used as a character in the string.

Q10 True or False: The following could be printed out following a RUN command: ONLY USE NUMERALS "1, 2, 3."

A10 False: Quotation marks cannot be used as a character in the string.

If you wish to change one or more statements in a program currently in the computer, you may do so *without* SCRatching the program and starting over. You merely type in a new statement, using the *same line number* as the line you wish to replace. Look at the program below, and the change made in it by replacing one line.

```
SCR              SCRATCH THE PRECEDING PROGRAM.
10 PRINT 7 + 5   ENTER THE NEW PROGRAM
99 END
RUN              RUN THE NEW PROGRAM.
12               HERE IS THE RESULT.
```

Next . . . *replace* Line 10 with a new Line 10. (*Replace,* means retype the line, beginning with the line number.)

```
10 PRINT 6*9
```

Now tell the computer to LIST the current program.

```
LIST

10 PRINT 6*9   HERE IS THE NEW LINE 10, AND THE
99 END         OLD LINE 99. AND THE OLD LINE 99.
RUN            RUN THE MODIFIED PROGRAM.
54             HERE IS THE NEW RESULT.
```

Q11 You may change or replace any line in your program by retyping it, using the same line number as the line you wish changed. The new statement _____ the old one with the same line number.

A11 replaces (or changes)

While we're on the subject, suppose you wish to take a statement out of a program *without* replacing it with another statement. *Don't* SCRatch and start over. Merely type the line number of the statement you wish deleted or removed and press RETURN.

```
10 PRINT 5+5
20 PRINT 12+3
30 PRINT 6+4
99 END   THIS PROGRAM IS IN THE COMPUTER, AND WE
         WISH TO DELETE (REMOVE) LINES 20 AND 30.
20       TYPE THE LINE NUMBERS ONLY, AND PRESS
30       RETURN AFTER EACH.
LIST     NOW, LIST THE PROGRAM.

10 PRINT 5+5
99 END   PRESTO! LINES 20 AND 30 ARE GONE.
```

Here is another program and a RUN of the program:

```
10 PRINT "MY COMPUTER
UNDERSTANDS ME"
20 PRINT "MY COMPUTER
CONFUSES ME"
99 END

RUN

MY COMPUTER UNDERSTANDS ME
MY COMPUTER CONFUSES ME
```

THIS OFFENDS US, SO WE WANT TO DELETE THE STATEMENT IN THE PROGRAM THAT CAUSED THE COMPUTER TO PRINT IT, AND WE WANT A RUN OF THE PROGRAM TO LOOK LIKE THIS:

```
RUN

MY COMPUTER UNDERSTANDS ME
```

Q12 What would you do to remove the offending statement from the program? _____

A12 Type 20 and press RETURN.

Q13 Show a LISTing of the program with the offending statement removed.

A13

```
LIST

10 PRINT "MY COMPUTER
 UNDERSTANDS ME"
99 END
```

NOTE: The quotation marks are included because this is a LISTing of the program itself, and not a RUN of the program.

When typing your programs into the computer, you may make a typing error or some other mistake. Look at this example.

```
10 PTINT 2*3+4
WE MISSPELL PRINT
SYNTAX ERROR
THE COMPUTER TELLS US WE MADE A MISTAKE. (SOME
COMPUTER SYSTEMS DO NOT INFORM YOU OF ERRORS
UNTIL YOU TRY TO RUN THE PROGRAM.)
```

The error message may be different on your computer. That's not the point. The point is, if you had noticed that you hit T when you meant to hit R, you could have immediately corrected your mistake by using the *back arrow* (←).

BEWARE! This method for correcting mistakes may not work on your computer.

The back arrow ← is on the same key as the letter O. To type a back arrow, hold the SHIFT key down and press the letter O.

This error correction information becomes quite obvious when operating a complete terminal. The Editors suggest you review use of the *back arrow* correction technique at the terminal.

Assume you just sat down at the computer terminal. You wish to know if there is a program currently in the computer's memory. Type LIST and then press the RETURN key. The computer will automatically type out the program (if there is one) that is in its memory. Here is an example.

```
LIST

10 PRINT "12 + 33"
99 END

THE COMPUTER AUTOMATICALLY TYPED ALL THIS.
```

Q14 The command which causes the computer to type out the program already in memory is _____.

A14 LIST

```
10 PRINT "12 + 33"
99 END
```

Assume that this program is currently in the computer's memory. Now you wish to add a new statement to the program, that says PRINT 12 + 33. You want the new statement to be evaluated by the computer *after* PRINT "12 + 33." The line number for the new statement must be greater than 10 and less than 99.

```
10 PRINT "12 + 33"
99 END
```

This program is stored in the computer. We type in the following statement:

```
20 PRINT 12 + 33
```

and then press the RETURN key. The new statement is then incorporated into the existing program. To verify this, type LIST, then press the RETURN key. The computer will type out the program with the new statement in line-number order. The computer will printout:

```
LIST

10 PRINT "12 + 33"
20 PRINT 12 + 33
99 END
```

If you type RUN with the preceding program in the computer, the computer would print:

```
RUN

12 + 33
 45
```

Q15 When you type LIST, in what kind of order will the computer type out the program including the latest corrections?

A15 In line order (or numerically)

If you retype Line 10 and added a comma to the end of the statement, the program would look like this when LISTed:

```
LIST
10 PRINT "12 + 33",
20 PRINT 12 + 33
99 END

RUN

12 + 33     45

NOTE THAT THE TWO RESULTS ARE PRINTED ON ONE
LINE.
```

Here is a variation of the program that causes the computer to print the problem (i.e., the string enclosed by quotation marks) and the answer on the same line.

```
10 PRINT "12 + 33 =",
99 END      12 + 33

RUN

12 + 33 =   45
```

Q16 In this program, fill in the blank to show what the computer would print.

```
10 PRINT "TWELVE PLUS
 THIRTY THREE EQUALS",
                12 + 33
99 RUN
RUN
```

A16 TWELVE PLUS THIRTY
THREE EQUALS 45

In BASIC, the comma and semicolon permit several expressions and/or strings to be printed on the same line. Look at the results of these two programs.

PROGRAM A

```
10 PRINT "12+33=", 12+33
99 END

RUN

12+33=        45
```

PROGRAM B

```
10 PRINT "12+33="; 12+33
99 END

RUN

12+33= 45
```

Examine the first statement in each program. Program A has a comma separating the string and the numerical expression, Program B has a semicolon.

Q17 The computer prints the results of the two parts of the PRINT statement *closer together* if you use a _____ instead of a _____.

A17 semicolon, comma

What You Have Learned

1. A computer terminal, such as a teletypewriter, provides a means for communicating with a computer. When a program is RUN (processed), the computer sends signals to the terminal which provide a visual output of the program. In this way, the terminal provides two-way communication between the computer and the user.
2. A computer program is typed a line at a time into the computer. Each separate line is considered a *statement* and begins with a *line number*. The computer reads and processes each statement consecutively depending on its line number. New statements may be added later, providing they may be labeled with line numbers consecutive with the older statements.
3. The computer responds to *commands*, which are words in the BASIC language. RUN commands the computer to process a program. LIST commands the computer to print out each statement of a program in its memory. SCR (Scratch) will command the computer to erase its memory (if this command is verified by hitting RETURN).
4. Quote marks when entered into a program command the computer to print out whatever is enclosed in the quotations, exactly as it appears. The information to be printed-out is referred to as a *string*. A string may contain numerals, letters or even special characters but it *cannot* contain quotation marks which may only enclose a string.

On most computers using BASIC, there are 5 standard *print positions* across a teletypewriter line. A comma in a PRINT statement causes the teletypewriter to move to the next available print position. For example,

```
10 PRINT 1, 2, 3, 4, 5
99 END

RUN

 ↑1        ↑2        ↑3        ↑4        ↑5
Position 1 Position 2 Position 3 Position 4 Position 5
```

Did you notice that the little arrows in the above example seem to be pointing to the space to the left of the number? This is where the print position actually begins. When the computer prints a positive number or zero, it prints a space first, then prints the digits of the number. Watch what happens when *negative* numbers are printed below positive numbers.

```
10 PRINT 1, 2, 3, 4, 5
20 PRINT -1, -2, -3, -4, -5
99 END.

RUN
 1         2         3         4         5
-1        -2        -3        -4        -5
 ↑         ↑         ↑         ↑         ↑
Position 1 Position 2 Position 3 Position 4 Position 5
```

Q18. Negative numbers are printed with a _____ followed by the digits of the number, while positive numbers are printed with a _____ followed by the digits of the number.

A18. minus sign (or negative sign, or "dash")
space

But what happens if there are *more than* 5 things in a PRINT statement? Watch.

```
10 PRINT 1, 2, 3, 4, 5, 6, 7, 8
99 END

RUN

 1         2         3         4         5
 6         7         8
```

The computer prints the 8 numbers on 2 lines with 5 numbers on the first line and 3 numbers on the second line.

Q19. What will the computer print during the following RUN?

A19.

```
10 PRINT 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
99 END


RUN

1         2         3         4         5
6         7         8         9         10
11        12
```

Hobby Computer Handbook

a "hard copy" of whatever appears on the CRT. The problem with this is that the CRT doesn't display a relatively large number of lines, so a long program or "run" will require many photographs or hard copies.

The printing terminal, of which a used model 33 teletypwriter is the most popular and least expensive (see article on TTY in this issue), provides a continuous print on a roll of paper. When the complete print is finished, the user simply tears off the printout. The only major problem with the TTY and similar terminals is that it's slow; about 110 wpm (words per minute) maximum, though a deluxe, much desired printing terminal, known as a DECWRITER operates at 110 or 300 wpm.

If you have money to spare, you can get a separate keyboard and a high speed printer. Some printers in the hobbyist price category print up to 80 characters per line, 150 lines per minute. If you want to get real fancy, modified IBM Selectric II terminals are available for hobbyist use, but make certain you get the correct model. Selectrics are available with EBCD and correspondence codes, and ASCII with RS-232 serial I/O. It's the ASCII model that's generally needed by hobbyists.

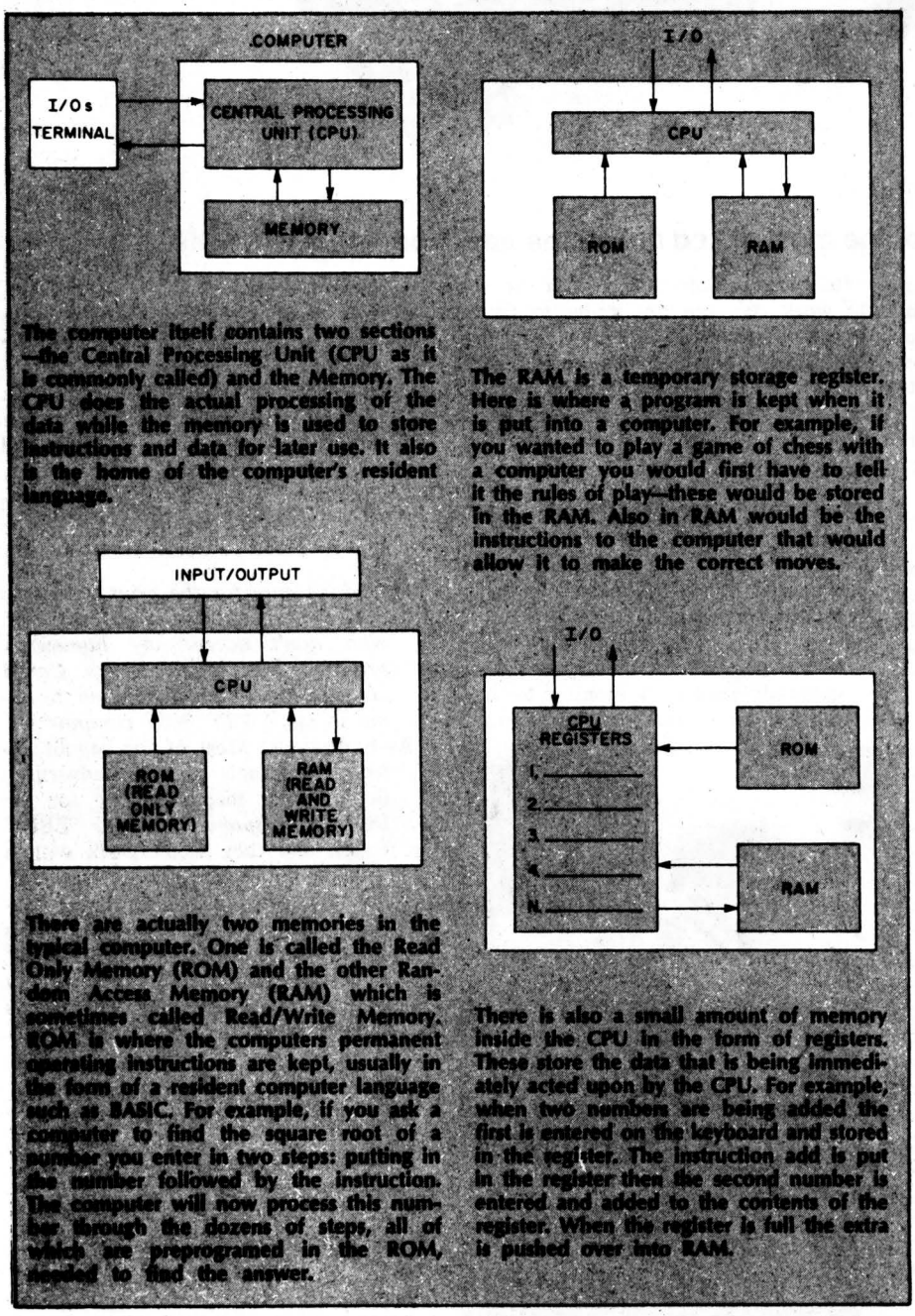
Here, shown in size relationship to each other, are two forms used for data storage; a mini-floppy disc, and a data cassette.

Peripherals. A terminal is called a "peripheral" device, meaning it's a piece of computer hardware that's external to the computer main frame. Besides a terminal, there are other peripherals. For example, a cassette recorder used for storing and loading data is a low cost peripheral. A full size (8-inch) or mini-disc (5¼-in.) mass data handling system is also a peripheral. So is an ordinary TTY punch that records computer data as a series of holes punched into a narrow tape.

As you've probably surmised, though it's possible to get started in hobby computing at nominal cost, a full-blown system complete with some form of data storage system is going to get expensive. Fact is, with sufficient memory to handle large jobs, such as keeping the records for a bowling league, and the necessary data storage system, figure on spending between $600 to $1000, more if you add a disc system.

Pays For Itself. The advantage to a full-blown system, however, is that it will probably serve as an excellent education aid for all ages—grade school through graduate studies – and with some imagination you can get the computer system earning money. Many hobbyists not only pay for their equipment, but also provide a steady sideline income, by using their computer for bowling league records, small business bookkeeping, and even fortune-telling at country fairs and *flea markets.* As you get more involved with hobby computing, you'll probably discover many new and exciting applications for your personal computer. Don't be afraid to think beyond the horizon to tomorrow. After all, who would have thought, even as late as 1977, that it would be possible for almost any hobbyist to own a full-blown computer.

(Continued on page 82)



Now check what happens when we use *semicolons* instead of commas to separate things in a PRINT statement.

```
10 PRINT 1, 2, 3, 4, 5
99 END

RUN

 1  2  3  4  5


10 PRINT 1; 2; 3; 4; 5; 6; 7; 8
99 END

RUN

 1  2  3  4  5  6  7  8
```

Semicolon spacing varies from computer to computer. The above RUNs show how our computer does it.

Q20. Things get printed closer together when we use a _____ instead of a comma.

A20. semicolon

Now let's see what happens when commas are used to separate two or more strings in a PRINT statement.

```
10 PRINT  "THIS","IS","CØMPUTER","PRØGRAMMING?"
99 END

RUN

THIS        IS          CØMPUTER    PRØGRAMMING?
```

In this PRINT statement, there are 4 strings, separated by commas. Each string is printed in a standard printing position. Here is a similar program using semicolons instead of commas.

```
10 PRINT  "THIS";"IS";"CØMPUTER";"PRØGRAMMING?"
99 END

RUN

THISISCØMPUTERPRØGRAMMING?
```

As you can see, with semicolon spacing, no spaces are printed between strings.

Q21. Using semicolons to separate two or more strings will:

1. Double space between strings
2. Insert no spaces between strings
3. Act the same as a comma

A21. (2) There will be no spaces inserted between strings. If you want spaces, include them in the strings.

```
10 PRINT  "THIS ";"IS ";"CØMPUTER ";"PRØGRAMMING?"
99 END
```

Q22. If we RUN this latest program, what will be printed?

A22.

```
THIS IS CØMPUTER PRØGRAMMING?
```

(Yes, this really is computer programming, although somewhat rudimentary. But read on!)

You have probably noticed that the plus (+) symbol of arithmetic tells the computer to add. The minus (−) symbol tells it to subtract. (It also indicates negative numbers.) The symbol for muliplication in BASIC is the asterisk (*), and the slash (/) is the symbol for division.

```
TØ TELL THE CØMPUTER TØ ADD, USE          +
TØ TELL THE CØMPUTER TØ SUBTRACT, USE     -
TØ TELL THE CØMPUTER TØ MULTIPLY, USE     *
TØ TELL THE CØMPUTER TØ DIVIDE, USE       /
```

Remember, when you want the computer to squeeze the answers or output more closely together, use semicolons instead of commas in the PRINT statement.

Here is a sample program to do simple arithmetic, with the results of a RUN of the program. Note the use of commas, and the widely spaced answers.

```
10 PRINT 7+5, 7-5, 7*5, 7/5
99 END

RUN

 12            2            35            1.5
```

Q23. Write a short program to do the following simple arithmetic. Group all of the expressions in one PRINT statement, using commas to separate expressions. Show the results you would predict for a RUN of your program, then try it on the computer if one is available.

$10 + 6$ $15 - 9$ $23 \div 5$ 3×13

A23.

```
10 PRINT  10+6, 15-9, 23/5, 3*13
99 END

RUN

 16            6            4.6           39
```

Here are some BASIC expressions in which two or more operations are used.

Expression	Value Computed by Computer
2*3 − 4	2
2 + 3*4	14

Q24. Here are three more BASIC expressions. What would be the values as computed by the computer after it does the indicated arithmetic?

(1) 2 * 3 + 4 * 5
(2) 2 + 3 * 4 − 5
(3) 2 * 3 − 4 * 5 + 6 * 7

A24. (1) 26 (2) 9 (3) 28

Now here are some more examples and exercises using division. The computer recognizes to do division when you program a slash (/).

Expression	Value Computed by Computer
3/4 + 5	5.75
2 − 3/4	1.25
2*3 + 4/5	6.8

Q25. Now give these division examples a try on your own.

(1) 3/4 + 5 * 6
(2) 2 − 3/4 + 5

A25. (1) 30.75; (2) 6.25

The computer does arithmetic in *left* to *right* order, with all multiplications (*) and/or divisions (/) performed *before* additions (+) and/or subtractions (−).

$\left.\begin{matrix}*\\/\end{matrix}\right\}$ before $\left\{\begin{matrix}+\\-\end{matrix}\right.$

Q26. Now try these. (*REMEMBER: Do arithmetic in left to right order.*)

Expression	Value Computed by Computer
2*3/4	_____
3/4*5	_____
3/4/5	_____
2*3/4+3/4*5	_____

A26. 1.5 Multiply 2 by 3, then divide result by 4.
3.75 Divide 3 by 4, then multiply result by 5.
.15 Divide 3 by 4, then divide result by 5.
5.25 First compute 2*3/4, then compute 3/4*5 then add the two results.

If you want to change the order use parentheses.

2*3 + 4 = 10
but 2*(3 + 4) = 14
2 + 3*4 + 5 = 19
but (2 + 3)*(4 + 5) = 45

Compute 3 + 4, then multiply result by 2.

Compute 2 + 3, then compute 4 + 5, then multiply those two results.

Q27. Complete the following. (*REMEMBER: Operations in parentheses are done first.*)

Expression	Value Computed by Computer
(2 + 3)/(4*5)	_____
2 + 3*(4 + 5)	_____
1/(3 + 5)	_____

A27. .25
.29
.125

One last look at the order in which arithmetic is done. In the expression below, the arrows in the circles show the order in which the operations are carried out. Write the final value for each expression.

Expression	Value Computed by Computer
⑤ ④ ③ ② ① 2 + 3 * (4 − (5 + 6 * 7))	−127
① ③ ② ④ ⑤ (3 * 4 + 5 * 6 − 7) * 8	4,375

Q28. Your next task is to write a correct BASIC expression to solve a given problem. Do so for each of of the following.

Remember to indicate all multiplaction and division operations with the proper BASIC symbol.

Problem	BASIC Expression
2 × 3 + 6 ÷ 7	_____
16(33 − 21)	_____
3.14 × 2 × 2	_____
88 − 52 18 + 47	_____

A28. 2*3 + 6/7
16*(33 − 21)
3.14*2*2
(88 − 52)/(18 + 47)

(Did you forget the asterisk?)

A BASIC program to compute and print the values of the expressions in the preceeding question would produce the following RUN:

```
RUN

 6.85714
 192
 12.56
 .553846
```

Q29. LIST the program which would produce the above RUN.

A29.

```
10 PRINT 2*3+6/7
20 PRINT 16*(33-21)
30 PRINT 3.14*2*2
40 PRINT (88-52)/(18+47)
99 END
```

There is a fifth arithmetic symbol in BASIC, which indicates raising a number to a power. This operation is called *exponentiation.*

↑ means raise to a power

For example,

Volume of a cube: V = S^3, where S is the length of a side.
If S = 5 and V = 5^3, then
V = 5^3 = 5 x 5 x 5 = 125.

Since a teletypewriter cannot print superscripts, you tell the computer to raise a number to a power by using the symbol

↑. On the teletypewriter, depress the SHIFT key and hold it while you press the ↑ N key.

```
10 PRINT 5↑3      (5↑3 means 5³ or 5 x 5 x 5)
99 END

RUN

 125
```

Q30. What would be the RUN for the following program?

```
          (2 ↑6 means 2⁶ or 2 x 2 x 2 x 2 x 2 x 2)

10 PRINT 2↑6
99 END

RUN
```

A30. 64

Here are the BASIC expressions for a couple more problems:

Problem	BASIC Expression
(a) $2^5 + 4^4$	2↑5 ÷ 3↑4
(b) $7 \times 7 \times 7 \times 7 \times 7 \times 7 \times 7$	7*7*7*7*7*7*7 or 7 ↑ 7

When evaluating a mixed expression of arithmetic operations, the computer computes powers (↑) *before* doing multiplication, division, addition, or subtraction.

The formula for computing the area of a circle is

$A = \pi r^2$

Let's use 3.14 as an approximate value of π and write a program to compute the area of a circle of radius 7.

```
10 PRINT  "IF RADIUS IS 7, AREA ØF CIRCLE IS"; 3.14*7 2
99 END

RUN

IF RADIUS IS 7, AREA ØF CIRCLE IS 153.86
```

In computing 3.14*7↑2, the computer first computes 7↑2, then multiplies that result by 3.14.

Computers use a special form of notation to indicate extremely large numbers, or extremely small decimal fractions. This method of expressing numbers is called *scientific notation*. Consider, for instance, a large number like the population of the earth which is about 3.6 billion people:

3.6 billion = 3 600 000 000

We asked *our** computer to print the population of the earth:

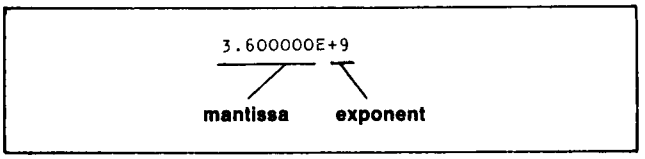
```
10 PRINT 3600000000
99 END

RUN

 3.600000E+9
```

**Your computer may do it somewhat differently.*

What's this?

Our computer printed the population of the earth in a form of *scientific notation*. (It really isn't especially scientific . . . it's just called that by some people.)

Scientific notation is simply a shorthand way of expressing very large or very small numbers. In scientific notation a number is represented by a *mantissa* and an *exponent*:



The mantissa and the exponent are separated by the letter E.

Here are some examples showing numbers written in good old every day notation and again in scientific notation (well, scientific notation according to our computer).

One trillion
- ordinary notation: 1 000 000 000 000
- scientific notation: 1.000000E+12

Volume of the earth in bushels
- ordinary notation: 31 708 000 000 000 000 000 000
- scientific notation: 3.170800E+22

Speed of a snail in miles per second
- ordinary notation: .0000079
- scientific notation: 7.9000000E-6

Q31. In each of the numbers expressed above what is the exponent? Is the exponent positive or negative?

A31. +12 positive; +22 positive; −6 negative

Have you noticed? Our computer always prints the mantissa with 7 digits, one digit to the left of the point, 6 digits to the right.

Numbers printed in scientific notation can be converted to ordinary notation as follows.

CASE 1. Exponent is positive.
- (1) Write the mantissa separately.
- (2) Move the decimal point of the mantissa to the RIGHT the number of places specified by the exponent. If necessary add zeros.

EXAMPLE: 6.123456E+4

4 places

(1) 6.123456 (2) 6.1234.56

Therefore, 6.123456E+4 = 61234.56.

EXAMPLE: 3.600000E+9

(1) 3.600000 (2) 3.600000000.

9 places (we had to add zeros)

Therefore, 3.900000E+9 = 3600000000.

Q32. Now you try it: 1.234567E+13

(1)_____ (2)_____

Therefore, 1.234567E+13 = _____

A32. 1.234567

1.2345670000000.

13 places (add 7 zeros)

12345670000000.

CASE 2. Exponent is negative.
- (1) Write the mantissa separately.
- (2) Move the decimal point of the mantissa to the LEFT the number of places specified by the exponent. If necessary, add zeros.

EXAMPLE: 7.900000E−6
(1) 7.900000 (2) .000007.900000
6 places (we added 5 zeros)
Therefore: 7.900000E−6 = .0000079

Q33. Now you try: 1.234567E−8
(1)_____ (2)_____
Therefore, 1.234567E−5 = _____

A33. 1.234567
.00001 .234567
5 places (we added 4 zeros)
.00001234567

What You Have Learned

1. On most computers using BASIC there are five standard *print positions* across a teletypewriter line. A comma in a PRINT statement moves the teletypewriter to the next available print position.
2. When you use a semicolon instead of a comma in a PRINT statement, strings get printed closer together. Thus, you can vary the spacing in the RUN of a program.
3. You can program a computer, in BASIC, by using the symbols +, −, * and / to mean, respectively, add, subtract, multiply and divide. The computer will do all arithmetic in *right* to *left* order, with all multiplications and/or divisions performed *before* additions and/or subtractions.
4. There is a fifth operation in BASIC called *exponentiation*. This function raises a number to a power and its symbol is: ↑ . For instance, 5 3 means 5^3 or $5 \times 5 \times 5$.
5. Computers use a special form of notation to indicate extremely large numbers, or extremely small decimal fractions–this is called *scientific notation*. A number in such notation is represented by a mantissa and exponent separated by the letter E. The number 1,000,000,000,000 would be represented on our computer by 1.000000E+12 in scientific notation.

SELF-TEST

If you can answer these questions, based on the first part of this article, you are a budding computer user and are ready to go on to the next part.

1. The device used to communicate programs to a computer is called a _____.
2. The individual lines of computer instructions in a program are called _____.
3. What is missing from this short program? _____

```
PRINT 2+2
END
```

4. Assume that you are at a computer terminal, typing a statement into the computer, and you notice that you have made a typing error. Describe a method of correcting your error (other than completely retyping the statement).
5. Describe a method for replacing a new statement for an old statement in a program without erasing the entire program and starting over.
6. Assume that there is a program in the computer. How do you erase that program from the computer's memory?
7. Assume that there is a program in the computer. How do you tell the computer to actually follow (or process) the program?
8. How do you cause the computer to type out a program stored in its memory?
9. Assume that this program is in the computer.

```
10 PRINT 3↑5
20 PRINT 8 3
99 END
```

Describe how to delete (remove) the second statement without erasing the entire program.

10. Write the symbols used in BASIC for the following arithmetic operations.
    addition _____
    subtraction _____
    multiplication _____
    division _____
    powers _____

Refer to this program to answer questions 11 through 18.

```
10 PRINT "MY CØMPUTER IS A WHIZ AT ARITHMETIC."
20 PRINT 5+2↑4 3
30 PRINT 8-16/32
40 PRINT (5+2)↑(8-3)
50 PRINT "THAT'S ALL, FØLKS!"
```

11. Which statements contain strings? _____
12. A string begins and ends with _____
13. Describe the order in which the computer does the arithmetic in Line 20.
14. Describe the order in which the computer does the arithmetic in Line 30.
15. Describe the order in which the computer does the arithmetic in Line 40.
16. In Line 40, why does the computer do the addition before the subtraction?
17. In general the computer does multiplication and division before addition and subtraction. Why is the order changed in Line 40?
18. Show what the computer will print when the program is RUN.
19. What symbol is used between several strings or expressions in a PRINT statement to cause the results to be printed close together when the program is RUN?

_____

20. Look at this program.

```
10 PRINT  10,20,30,40,50,60,70
99 END
```

How many *lines* will the results of RUNing the program occupy? _____

21. Convert the following numbers from scientific or "E" notation into standard notation.

Scientific Notation	Ordinary Notation
1.123456E+6	_____
1.123456E+12	_____
7.777777E−2	_____

1.000000E−12

22. Write a computer program to do the following arithmetic and produce the results shown in the RUN below.
    (a) 10^8
    (b) 10^{12}
    (c) $18.56 - 9.35$
        $2.12 + 3.3^3$

```
RUN

TEN RAISED TØ THE 3RD PØWER = 1000
TEN RAISED TØ THE 12TH PØWER = 1.000000E+12
THE ANSWER TØ PRØBLEM (C) IS 1.689908
```

Answers to Self-Test

1. Computer terminal (e.g., teletypewriter).
2. Statements
3. Line numbers
4. Type a black arrow (←) to erase each character (right to left) that you wish deleted until the mistake is erased. Then finish typing the statement, beginning at the point where the error was made.
5. Using the line number of the statement you wish replaced, type in the new statement.
6. Type SCR (for SCRatch), and press the RETURN key.
7. Type RUN and press the RETURN key.
8. Type LIST and press the RETURN key.
9. Type the line number of the line to be deleted (20) and press the RETURN key.
10. + addition
    − subtraction
    * multiplication
    / division
    ↑ powers
11. Lines 10 and 50.
12. Quotation marks
13. ↑, *, +. First the computer computes 4↑3, multiplies the result by 2, then adds 5.
14. /, −. First the computer divides 16 by 32, then subtracts the result from 8.
15. +, −, *. First the computer add 5 and 2, next it subtracts 3 from 8, and then it multiplies the two results.
16. The computer does the operations contained in parentheses in left to right order.
17. The computer does arithmetic contained in parentheses first.
18.

```
RUN

MY CØMPUTER IS A WHIZ AT ARITHMETIC.
 133
 7.5
 35
THAT'S ALL, FØLKS!
```

19. Semicolon.
20. Two lines, like this:

```
10        20        30        40        50
60        70
```

21.

```
1.123456E+6          1123456.
1.123456E+12         1123456000000.
7.777777E-2          .07777777
1.000000E-12         .000000000001
```

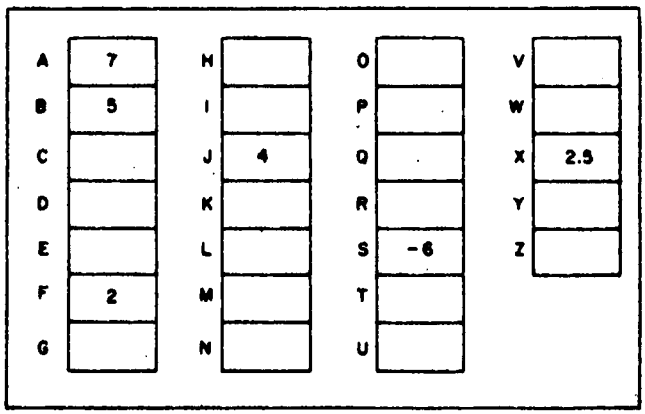
22.

```
10 PRINT "TEN RAISED TØ THE 3RD PØWER =";10 3
20 PRINT "TEN RAISED TØ THE 12TH PØWER =";10 12
30 PRINT "THE ANSWER TØ PRØBLEM (C) IS";(18.56-9.35)/(2.12+3.33)
99 END
```

To illustrate the concept of *variable* and the function of the LET statement in BASIC, imagine that there are 26 little boxes inside the computer. Each box can contain one number at any one time:

We have already stored numbers in some of the boxes. For example,

7 is in box A
5 is in box B



Q1. (a) What number is in box F?
(b) In box J?
(c) What box is −6 in?
(d) What box contains 2.5?

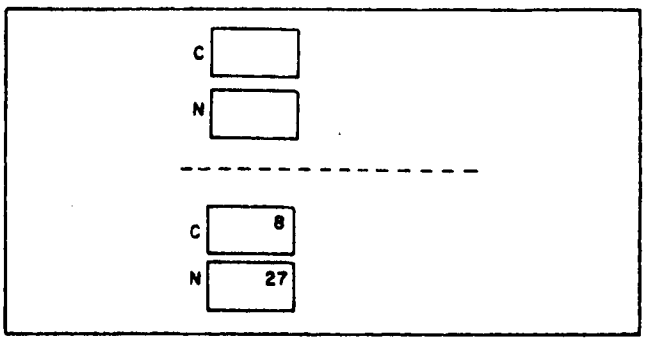
A1. (a) 2, (b) 4, (c) S, (d) X

Boxes C and N are shown again below. Use a *pencil* to do the following.

(a) Put 8 into Box C. In other words, write the numeral "8" in the box labeled "C."

(b) Put 12 into N.

(c) Put 27 into N. But wait! A box can hold only one number at a time. Before you can enter 27 into N, you must first erase the 12 that you previously entered.



When the computer puts a number into a box, it *automatically* erases the previous content of the box, just as you did. In order to put "27" into Box N, you first erased the previous content, "12."

We call A, B, C, . . . , Z *variables*. The number in Box A is the *value of A;* the number in Box B is the *value of B;* the number in C the *value of C* and so on.

Below is a program that uses the LET statement to instruct the computer to "put a number in a box," or more technically to assign a numerical *value* to a *variable*. This program tells the computer to

```
10 LET A=7  ← Put 7 into Box A.
20 PRINT A  ← Print the content of Box A
99 END
RUN

 7
```

Q2. (a) What is the *variable* in Line 10?
(b) What *value* is assigned to that variable?

A2. (a) A, (b) 7

Q3. Complete the following program to assign the value 23 to the variable X and then print the value of X.

```
10 .....
20 .....
99 END
```

A3. *(Answer is listed below.)*

```
10 LET X=23
20 PRINT X
99 END
RUN

 23
```

Here is another example. This program adds four numbers, which might be scores of some kind, and computes the mean (average).

```
10 LET A=5
20 LET B=8
30 LET C=3
40 LET D=6
50 PRINT "SCØRES:";A;B;C;D
60 PRINT "MEAN:";(A+B+C+D)/4
99 END
RUN

SCØRES: 5  8  3  6
MEAN:  5.5
```

The LET statements in this program tell the computer to ssign numerical values to variable, in this case to put values , 8, 3 and 6 into boxes A, B, C, and D. These values are rinted (Line 50) and then the computer uses them (Line 0) to compute and print the mean.

Q4. Complete each of the following RUNs as you think ie computer would do it. If possible, use a computer to find out if you are correct.

```
10 LET A=1
20 LET A=2
30 PRINT A
99 END
RUN
```

```
10 LET A=7
20 LET B=A
30 PRINT B
99 END
RUN
```

```
10 LET A=1
20 PRINT A
30 LET A=2
40 PRINT A
99 END
RUN
```

A4. (a) 2

Note that the second value assigned to A in Line 20 replaced the value assigned to A by Line 10.

(b) 7
(c) 1
(c) 2

Look at the preceeding programs (a), (b) and (c). In program (b) the value of one variable is used to assign a value to another variable!

So it turns out that one variable can take its value from another variable. Not only that, but a variable can get its value from *computations* involving one or more other variables *whose values have been previously assigned.* (That last part is important.)

We can illustrate this with a program that will calculate the grade point average for a student. Assume the student received:

4 units of A
6 units of B
4 units of C
2 units of D
0 units of F

```
100 REMARK GRADE PØINT AVERAGE PRØGRAM USING LET STATEMENTS
110 LET A=4
120 LET B=6
130 LET C=4
140 LET D=2
150 LET F=0
160 LET U=A+B+C+D+F
170 LET G =(4*A+3*B+2*C+1*D)/U
180 PRINT "YØUR GRADE PØINT AVERAGE IS";G
999 END

RUN

YØUR GRADE PØINT AVERAGE IS 2.75
```

Q5. In Line 160, U (for Units) receives it's value from the total of each letter grade. What numerical value does U receive when this program is RUN?

A5. U=16

Q6. Which line of the program computes and assigns the *computed value* to the variable G?

A6. Line 170

LET statements are all fine and good, but what a hassle to change all those LET statements in Lines 110 to 150 everytime you want to calculate the GPA (Grade Point Average) for a different set of grades. Ah, but leave it to BASIC to come up with a clever solution – namely the INPUT statement.

The INPUT statement allows the computer user to assign *different* values to INPUT variables each time a program is RUN *without* modifying the program itself. When the com-

puter comes to an INPUT statement in a program, it types a question mark and waits for the user to enter a value for the INPUT variables (or variable). Here is an example.

```
20 INPUT A
30 PRINT "THIS TIME A =";A
99 END

RUN
```

In our example, we typed in 3 as the value to be assigned to A, pressed RETURN, and the computer then continued running the program, using A = 3. Here's the program again with the completed RUN:

```
20 INPUT A
30 PRINT "THIS TIME A =";A
99 END
RUN

?3
THIS TIME A = 3
```

The *value* of A is printed after the string

After we typed RUN and pressed the RETURN key, the computer typed a question mark. We then typed a 3, which is our value for the INPUT variable A. The computer then printed the string THIS TIME A = followed by the numerical value of A.

Q7. The program could be run again with a different value of A supplied by the user. What would a RUN look like if the user typed 7 as the value of A?

A7. *(Answer is listed below.)*

```
RUN

?7
THIS TIME A = 7
```

Now, in order to make things really clear when dealing with INPUT statements, we need a way of informing the user what the INPUT statement is asking for. Let's add this statement to our example program:

```
10 PRINT "WHAT IS YØUR VALUE FØR A";
```

See the semicolon at the end of the PRINT statement?

```
10 PRINT "WHAT IS YØUR VALUE FØR A";
20 INPUT A
30 PRINT "THIS TIME A =";A
99 END

RUN

WHAT IS YØUR VALUE FØR A?
```

This much comes from Line 10

The question mark comes from the INPUT statement in Line 20

When a semicolon is used at the end of a PRINT statement, the teletype stays on the same line *instead of* performing a "carriage return" and going to the beginning of the next line. Here is our revised program, and the beginning of a RUN.

Now we know exactiy what the computer is waiting for — a value for the variable A. We use 350 as the value, type it in after the question mark, then press RETURN.

```
RUN

WHAT IS YØUR VALUE FØR A?350
THIS TIME A = 350
```

Here's another RUN of the program, but now the user has entered 17 as the value of A.

```
RUN

WHAT IS YØUR VALUE FØR A?17
THIS TIME A = 17
```

Q8. Now you do one. Write a program, using *two* INPUT statements, that will result in the following printout when RUN.

```
RUN

VALUE ØF X?5          Values supplied by user
VALUE ØF Y?10
THEN X + Y = 15       Value computed
```

A8. Either of these two programs is correct.

```
10 PRINT "VALUE ØF X";
20 INPUT X
30 PRINT "VALUE ØF Y";
40 INPUT Y
50 PRINT "THEN X + Y =";X + Y
99 END
RUN

VALUE ØF X?5
VALUE ØF Y?10
THEN X + Y = 15
```

```
10 PRINT "VALUE ØF X";
20 INPUT X
30 PRINT "VALUE ØF Y";
40 INPUT Y
50 LET Z = X + Y
60 PRINT "THEN X + Y =";Z
99 END
RUN

VALUE ØF X?5
VALUE ØF Y?10
THEN X + Y = 15
```

So much for theory. Now let's apply the capabilities of th INPUT statement to the Grade Point Average program.

```
100 REMARK PRØGRAM TØ CØMPUTE GRADE PØINT AVERAGE
110 PRINT "HØW MANY UNITS ØF A";
120 INPUT A
130 PRINT "HØW MANY UNITS ØF B";
140 INPUT B
150 PRINT "HØW MANY UNITS ØF C";
160 INPUT C
170 PRINT "HØW MANY UNITS ØF D";
180 INPUT D
190 PRINT "HØW MANY UNITS ØF F";
200 INPUT F
210 LET U=A+B+D+F
220 LET G=(4*A+3*B+2*C+1*D)/U
230 PRINT
240 PRINT "YØUR GRADE PØINT AVERAGE IS";G
999 END
```

A PRINT statement with nothing following it causes the teletype to advance to the next line without printing anything, leaving "line spaces" as you'll see in the next RUN.

Here is a RUN of the preceding program featuring values of A, B, C, D and F supplied by the user.

```
RUN
HØW MANY UNITS ØF A?4
HØW MANY UNITS ØF B?6
HØW MANY UNITS ØF C?6
HØW MANY UNITS ØF D?O
HØW MANY UNITS ØF F?O

YØUR GRADE PØINT AVERAGE IS 2.875
```

Following each question mark, the user typed the requested value, then pressed the RETURN key

After all 5 values had been entered, the computer computed the GPA and printed it

Q9. (a) How many units of A did the user enter?
(b) How many units of F?
A9. (a) 4 (b) O

Let's demonstrate another capability of the INPUT statement. One INPUT statement can be used to assign values of *two or more* variables:

```
10 PRINT "VALUES ØF X AND Y";
20 INPUT X,Y
30 PRINT "THEN X + Y =";X + Y
99 END

RUN

VALUES ØF X AND Y?12,6
THEN X + Y = 18
```

There are two things to note:

```
20 INPUT X,Y    ← No comma after the last variable
        ↑  ↖
No comma here   Comma separates the variables

RUN

VALUES ØF X AND Y?12,6  ← No comma after last value
                  ↑
            Comma between values
```

Note that 12 is the value assigned to the first INPUT variable X, and 6 will be assigned to the second INPUT variable

Here is the summary
When a program containing an INPUT statement with ultiple variables is RUN, the first value typed in by the user er the INPUT question mark will be assigned to the first riable that appears in the INPUT statement; the second lue typed in by the user will be assigned to the second variable appearing in the INPUT statement, etc. Both the variables in the INPUT statement in the program, *and* the values typed in by the user when the program is RUN, must be separated by commas.

Here is another RUN of the program in frame 14. We want to enter 73 as the value of X and 59 as the value of Y.

```
RUN

VALUES ØF X AND Y?73
?
```

Whoops! We absent-mindedly hit the RETURN key

The computer typed another question mark. This means "Didn't you forget something?"

We then completed the RUN by entering the *second* number, the value of Y. Here is the complete RUN:

```
RUN

VALUES ØF X AND Y?73
? 59
THEN X + Y = 132
```

If we don't enter a numerical value for *every* variable in an INPUT statement, *our* computer types a question mark.

Q10. Your turn. Write a program to compute and print the value of A*(B+C) for INPUT values of A, B, and C. A RUN should look like the following.

```
RUN

VALUES ØF A,B,C?2,3,4
THEN A*(B + C) = 14
```

A10. Here are two ways to do it.

```
10 PRINT "VALUES ØF A,B,C";
20 INPUT A,B,C
30 PRINT "THEN A*(B + C) =";A*(B + C)
99 END
```

```
10 PRINT "VALUES ØF A,B,C";
20 INPUT A,B,C
30 LET D=A*(B + C)
40 PRINT "THEN A*(B + C) =";D
99 END
```

Here's a version of the program to calculate Grade Point Average that uses only one INPUT statement to tell the computer how many units of A, B, C, and F you received (or expect to receive). It uses a PRINT statement before the INPUT statement in order to identify the INPUT values needed.

Learning About Computers

```
100 REMARK PRØGRAM TØ CØMPUTE GRADE PØINT AVERAGE
110 PRINT "UNITS ØF A,B,C,D AND F";
120 INPUT A,B,C,D,F
130 LET U=A+B+C+D+F
140 LET G=(4*A+3*B+2*C+1*D)/U
150 PRINT
160 PRINT "YØUR GRADE PØINT AVERAGE IS";G
999 END
```

You may have noticed the REMARK statement used as a heading for various example programs. That's what it is, a remark by the programmer to identify what a program or a section of a program does. REMARK statements exist solely for the convenience of a person looking at a program, and (for a change) don't tell the computer to do anything. We will use REMARK to identify most of the programs that follow.

Q11. What will a RUN of the program look like if we enter 2 units of A, 5 units of B, 4 units of C, 3 units of D and 3 units of F?

A11. *(Answer is listed below.)*

```
RUN

UNITS ØF A,B,C,D AND F?2,5,4,3,3

YØUR GRADE PØINT AVERAGE IS 2
```

Now, let's consider a problem in the field of population growth.

PROBLEM: In year zero, we start with a population of P people. The population increases by 1% each year. In N years, what will the poulation be?

P is the initial population.
R is the growth rate in percent per year.
N is the number of years.
Q is the population after N years.

$$Q = P(1 + 1/100)^N$$

1% increase per year
N years
Initial population
Population at the end of N years

If the growth rate is 2.5% per year, then

$$Q = P(1 + 2.5/100)^N$$

And, if the growth rate is R% per year, then

$$Q = P(1+r/100)^N$$

Q12. For review, write this last formula as a LET statement for variable Q using BASIC notation.

170 LET Q = ...

A12. *(Answer is listed below.)*

```
170 LET Q=P*(1 + R/100) N
```

NOTE: This formula may actually be used to compute the growth rate for anything that increases by a fixed proportion percentage for a given length of time (e.g., interest on money, bacteria culture growth, etc.).

Here is one version of a population growth program.

```
100 REMARK PRØGRAM TØ CALCULATE PØPULATIØN GRØWTH
110 PRINT "INITIAL PØPULATIØN";
120 INPUT P
130 PRINT "RATE ØF GRØWTH";
140 INPUT R
150 PRINT "NUMBER ØF YEARS";
160 INPUT N
170 LET Q=P*(1+R/100) N
180 PRINT
190 PRINT "PØPULATIØN AFTER";N;"YEARS IS";Q
999 END

RUN

INITIAL PØPULATIØN?1000
RATE ØF GRØWTH?1
NUMBER ØF YEARS?20

PØPULATIØN AFTER 20 YEARS IS 1220.19
```

Q13. It is now the year 1973. The population of the eart is about 3.6 billion people. The growth rate is about 2% pe year. Suppose this growth rate persists until the year 200 We want to know what the population will be in 2001. Sho how this information is entered by completing the blanks i the following part of a RUN.

```
RUN

INITIAL PØPULATIØN?
RATE ØF GRØWTH?
NUMBER ØF YEARS?
```

A13. We think you did it this way.

```
RUN

INITIAL PØPULATIØN?3600000000
RATE ØF GRØWTH?2
NUMBER ØF YEARS?28

PØPULATIØN AFTER 28 YEARS IS 6.267686E+09
```

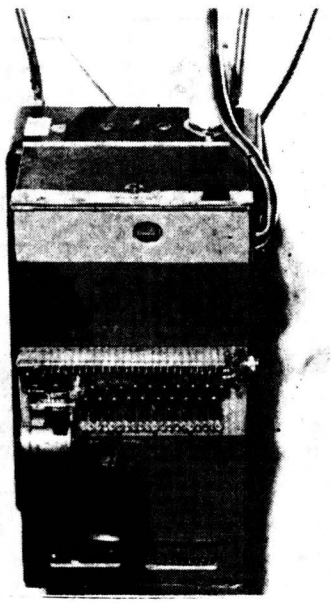
Or perhaps this way:

```
RUN

INITIAL PØPULATIØN?3.6E9
RATE ØF GRØWTH?2
NUMBER ØF YEARS?28

PØPULATIØN AFTER 28 YEARS IS 6.267686E+09
```

(Continued on page

Hearing Again Through Microprocessors

Electronics converts sound into feeling for the totally deaf.



The closeup on the upper left shows the reverse side of the Teletactor's belt, with its tiny stimulators that vibrate against the wearer's skin. As they vibrate in different patterns corresponding to varying sounds, even separate words can be distinguished with some practice. On the lower left, a deaf person models the "tickle belt" while holding the microphone in his hands. Under normal use the belt would be hidden by the individual's clothes. The battery-powered microphone and electronic processor is shown above. This unit receives sound impulses, converts them to electrical pulses, and passes them to the belt, which separates them into individual vibrations. The Teletactor allows its users much greater daily freedom.

☐The Teletactor is a new device which promises a limited sort of "hearing" to the deaf. A deaf patient wears a belt around his waist, a special belt which causes a tickling sensation on the skin when sounds are produced.

"Patients really can't converse with our belt," said Dr. Frank Saunders, one of the belt's developers, "but they are able to recognize a number of situations." The totally deaf, according to Saunders and others at the San Franciscan Smith-Kettlewell Institute of Visual Sciences, can be taught a new, limited kind of "language."

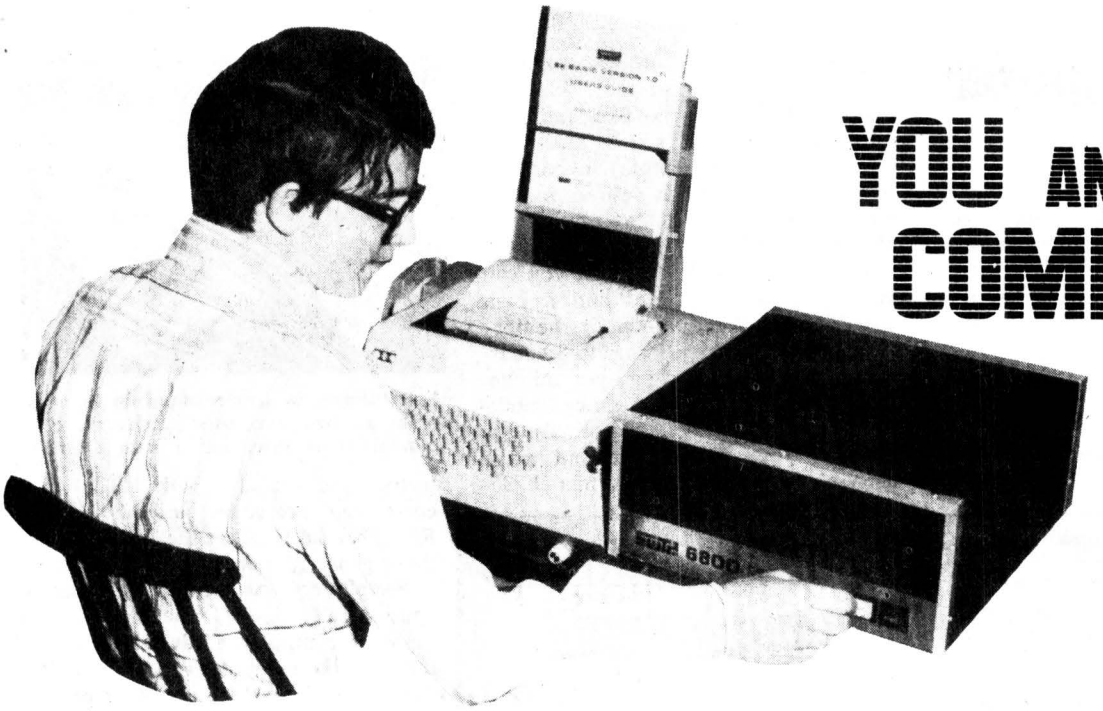
A deaf patient would be able to "hear" (actually, to feel) the tea kettle boiling or the doorbell chiming, or even the ringing of a telephone. Saunders maintains such informational input would be of great use to the deaf, though at first glance it might seem that it is of little use to a deaf individual to know the phone is ringing as they cannot hear the caller.

"This is so only up to a point," Saunders said. "If the caller wants to know how the patient is and is informed about the belt, he can use a world like 'allright' the sound of which will be instantly translated to a tickle and the patient will get the message. The patient could also inform emergency services in case of fire and say, 'There's a fire here. If you are getting my message, say yes. If he hears a response, the patient could then leave his address."

Actually, with the newest model of the Teletactor, patients have learned to distinguish many different words such as: I, you, we, he, want, like, see, and other simple one-syllable or evenly accented words. Once the deaf person becomes used to the device, his "vocabulary" will increase proportionally.

The device is lightweight, and simple to operate. The belt about the waist is connected to a battery-powered mike worn outside the clothing. All manner of sounds, from speech to loud noises are converted into electrical impulses of varying strengths. The inner side of the belt is dotted with twenty 'stimulators.' These vibrate against the patient's skin.

Right now, the Teletactor is an expensive aid, priced at approximately $400. It is expected, however, that the price will drop significantly in the near future. More information on the tickling belt can be had by writing to Dr. Frank Saunders, c/o the Smith-Kettlewell institute of Visual Sciences, San Francisco, Calif. 94115. ■



YOU AND YOUR COMPUTER

Answers to some of the most asked questions about personal computers

IT'S BEEN LESS THAN TWO years since we heard the first rumors that both Heathkit and Radio Shack were working on personal computers for the electronic hobbyist, yet in the few short months since the rumors were proved correct *personal computing* has become the hottest thing going for the electronic hobbyist and experimenter.

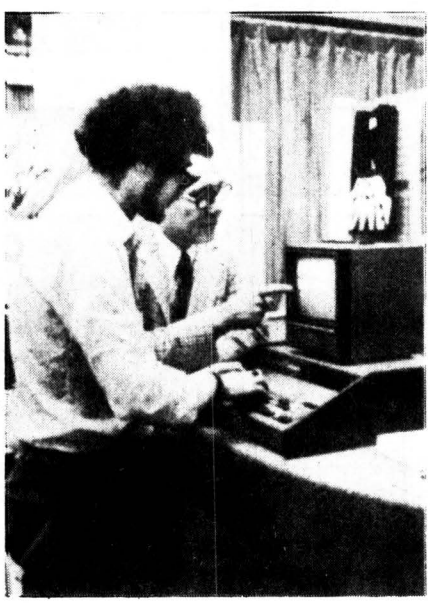
It has also become the most confusing, with each manufacturer and distributor inventing new terminology to prove, or imply, his computer, system, or accessory is the best. Even trained computer and data-handling experts with advanced degrees in computer science are often at a loss to explain what in heck many computer dealers are talking about. Personal computing has become a Tower Of Babel; and as yet there is no Rosetta Stone the average hobbyist can use to unscramble *computerese*—a foreign language even more complex than French.

Because the HOBBY COMPUTER HANDBOOK editorial staff has such extensive experience in choosing, setting up, using and maintaining microcomputers for home and hobby projects, we are regularly deluged with questions about computers. Because we can talk for hours about computers and computer accessories, what's new and what we can expect in the next few years, we thought we'd share our knowledge with you in the clearest, most efficient form, the direct question and answer.

For some, the answers will appear simplistic; but keep in mind we are trying to avoid *computerese*. Our primary purpose is to provide you, our reader, with concrete information you can put to work. We are not going to try to impress you with anyone's expertise. We know computer equipment represents a substantial investment so we aim to present our information in the most useful manner—and that means straight English.

Question—*Since some computer kits are priced almost the same as complete computers having built in BASIC and a keyboard, what is the advantage in building a kit?*

Answer—The complete computers such as the Apple, OSI, and Radio Shack generally need a TV monitor for output display, or the computer's output signal is fed through an *RF Modulator* to an ordinary TV set which serves as the display. At present, the complete computers have no peripherals for providing "hard copy" (a printout). The computer kits, on the other hand, permit almost unlimited expansion, though it does get costly when you add in the cost of a complete terminal: either TTY (teletype), CRT, or Selectric typewriter.



Processor Technology manufactures the SOL-20, shown here. The SOL is based on the 8080 chip, has a self-contained keyboard and utilized the S-100 bus. It has, in the smallest version, 8K of memory and BASIC in ROM. Available in kit or wired.

Q—*I don't care for the print quality of a TTY terminal, nor do my teachers, who don't accept my homework printed all in capital letters. Can I connect a Selectric typewriter terminal to my SWTP 8600 computer?*

A—It depends. Most of the rebuilt Selectric terminals you find advertised in computer magazines are for the IBM *correspondence* and EBDC codes, and they won't work with a personal computer. Some surplus outfits, however, build in an ASCII converter with an RS-232 output. If you can find one of these ASCII/RS-232 Selectrics (about $900) you simply connect it to a personal computer's serial RS-232 I/O. (Just plug it in.)

Q—*I would like to get my child started in computing. Which of the beginner's kits in the $100 range do you suggest?*

Computer Supermarkets

(Continued from page 27)

an advanced hobbyist(s) who guides you into your first purchase, and it will be experimenters like yourself and perhaps advanced hobbyists who will give you fresh ideas on expanding your computer installation. Unless the store creates an atmosphere welcoming hobbyists like yourself, providing a place to get together to try new equipment and talk it out, you'll be floundering all alone in a sea of computer equipment —and it can get very expensive that way. You can wind up with a lot of hardware you don't need or can't use.

Summing Up. We have several computer systems in operation throughout our staff, and in every instance we were going nowhere at all until getting "professional" guidance: It's one thing to read that we must "initialize the computer," it's quite a different thing to have someone tell us to simply "set the pointer to memory address 0100."

To be frank, an awful lot of the instruction manuals which we have seen supplied with the equipment we have handled is abysmal. You're going to need a lot of free assistance, particularly in the way of concise operating instructions, so be certain you pick a store capable of meeting your equipment needs while providing the atmosphere for meeting the hobbyists that can help you get your system up and running with the least amount of difficulty.

In short, many of you will need a lot of assistance at the beginning, so you had better select a computer store willing and capable of meeting your personal requirements.

Finally, and most important if you're a newcomer or beginner at personal computing, the store must want you. That might seem like a simple enough idea, but the truth of the matter is that many computer stores are gathering places for hobbyists, where strangers get together to swap ideas, programs, or just to learn "a little bit more." Well, people hanging around aren't welcome everywhere. We've been in computer stores with just two chairs, one for the salesman and one for the customer. In a gathering place, such as the Computer Mart of New York, you can hang around, try different systems, ask questions, and no one keeps asking "Can I help?" over and over until you get the idea it's either buy or leave. ■

Bio-Logic

(Continued from page 48)

letters is presented to you. Could you memorize it at one pass?

VTVMSCRFETUJT

You could if you treated the items like this:

VTVM SCR FET UJT

This type of organization is called chunking, which means ordering the list into meaningful groups. Since meanings are governed by the pattern of storage in LTM, previously acquired LTM information is being used here to facilitate STM storage. The chunked list has four items, well within the memory span, and can be quickly memorized by anyone familiar with electronics. But to an uninitiated person the list contains thirteen items; a few rehearsals would be necessary to enable perfect recall.

Let's try another experiment. Can you memorize the following list quickly and without a lot of rehearsing?

110001100101000011

If you can count in binary, it's a cinch. Look at the list in this way:

110 001 100 101 000 011

6 1 4 5 0 3

Below each triplet is its decimal equivalent. If you can mentally do such a conversion, you're left with only six decimal digits to memorize. When the time comes for recall, simply change the decimal numbers you've memorized back to binary. With some practice, you should be able to handle lists of about twenty-one ones and zeros, much to the amazement of your friends. Note that you're using old LTM information —the rules of binary-to-decimal and decimal-to-binary conversion—to assist in memorizing new material.

RAM. Leave STM now and consider long-term storage. First of all, think of all the different kinds of information contained in LTM. Sights, sounds, tastes, smells, and words are all stored in some codified form. The information can be reached quickly without first searching through a lot of extraneous material. In electronics we call such a memory structure random-access. This random-access characteristic is one of LTM's most important features, and current theories hold that a close relationship exists between the structure of LTM and the grammar of language.

Certainly one of the most important characteristics of LTM is its relative permanence of storage. However, we all know that recall is often marred by forgetting. Is this a memory failure? It might be, but some say no. Consider the experiences of the neurosurgeon Wilder Penfield. While performing surgery he electrically stimulated areas of patients' brains. Since the patients were conscious during brain surgery (there are no pain receptors in the brain), they were able to report their sensations. Many reported long-since forgotten, seemingly trivial incidents from the past. These sensations were vivid, filled with minute visual and auditory details. In fact, they were often described as being like movies. This type of evidence is frequently cited in support of the notion that everything stored remains stored, and that forgetting is the inability to find stored information.

There is no question, however, about the role of the chunking process in LTM. Chunking is the most effective method of guaranteeing long-term storage. Tests on mnemonists, people with exceptional powers of memory, show conclusively that structuring information in some imaginative way is the key to successful memorization. For instance, given a list of words to memorize, the mnemonist might weave them together into a fanciful story.

So far we've only scratched the surface of the topic of human information-processing, and in an article of this size we can't do much more than that. Many questions remain unresolved. ■

Programming with Basic

(Continued from page 78)

1. A computer uses a LET statement to assign a numerical *value* to a *variable*. One variable can take its value from another variable. A variable can also get its value from *computations* involving one or more variables *whose values have been previously assigned.*
2. The INPUT statement allows the user to assign *different* values to INPUT variables each time a program is RUN *without* modifying the program itself. When the computer comes to an INPUT statement in a program, it types a question mark and waits for the user to enter a value for the INPUT variable(s).
3. An INPUT statement allows for multiple variables. The first value the user types after the question mark produced during a RUN will be assigned to the first variable in the INPUT statement; the second to the second variable; and so forth. All values typed in by the user must be separated by commas.
4. REMARK statements exist solely for the convenience of a person examining the program. They do not tell the computer to do anything. They can be used as headings, to label checkpoints, anything the programmer desires.

TI-58 Calculator

(Continued from page 41)

covering such diverse specialties as aviation, marine navigation, real estate, and investment.

Big Brother. The TI 58 is available in an even more elaborate version, the TI 59. The TI 59 has exactly twice the step/memory capability of the 58; it allows up to 960 program steps or 100 memories. To give you an idea of just how sophisticated this is, the TI 59 has 715 kilobytes of ROM, 1 kilobyte of RAM and a 70 usec. instruction time.

The TI 59 also offers a non-volatile memory, magnetic cards. The cards can be recorded on right from the keyboard after being fed into a slot on the side of the TI 59. Once recorded, a program can never be lost to you (unless you misplace the mag-card!). All you have to do is to plug the card back into the calculator and run the program.

Minding The P's and Q's. If all this weren't enough, TI has carried their computer concept a step further and provided for direct printout so you can get a listing of a program, or a run. The TI-58 (and the TI-59) can be attached to a model PC-100A alphanumeric printer/plotter. No we haven't made a mistake, we mean alphanumeric. Though the basic calculator is only numeric, when attached to the PC-100A (not the PC-100) you can obtain a letter print by keying corresponding numerals on the calculator. This permits you to identify programs and steps in English, rather than a symbol code. The PC-100A also can plot results from a calculator program instead of simply printing a list of values. If this *still* doesn't make you think of these calculators as computers, consider that with the PC-100A, user-prompting can be entered into the program. The printer can thus announce when it is time to enter a variable during a program's run. This allows untrained people to use a previously prompt-prepared program.

Add all these features together—they total *computer*. The printer attachment might be somewhat expensive for many pockets, but even without it you get essentially all the computer features even in just the TI-58.

Both the TI-58 and TI-59 come complete with NiCad batteries, a charger, a *Master Library* module with 89-page contents guide, the calculator instruction manual, which is a complete course in programming, and a belt-loop softcase. The 59 also comes with mag cards. Price for the TI-58 is $124.95, and $299.95 for the TI-59.

Apple II

(Continued from page 17)

had two samples) using either a real cheap recorder ($30), or a better quality (about $90) model. Unusually, we could not load BASIC using the *moderately* priced machine. Apple specifically recommends a particular model, a moderately priced machine which we could not obtain locally; so our advice when looking for a recorder to go with the Apple II is to try before you buy.

Tasting The Apple. Most likely because of the color graphics, games are an important part of Apple II's orientation. The computer is supplied with two paddles, wired together, which connect to the computer by simply plugging each into what would otherwise be a standard IC socket. Provision is made for future expansion through prewiring of the game I/O port for four paddles, 3 TTL inputs, and 4 TTL outputs.

Depending on your technical knowledge and sophistication, our description of the Apple II might appear too simple or too complex. This is so because the Apple II can be used on two different levels. Loaded with the 16K Applesoft BASIC it is an unusually easy to use, extremely powerful personal computer suitable for the student from elementary grades clear through college and advanced studies. Alternately, the Apple II has sufficient complexity and adaptability for the computer technology student and experimenter who wants to do a lot of homebrew expansion and adaptation. It's sort of like the movies made back in the thirties; a cute plot for the kiddies, and slice-of-life for the adults, with each patron seeing only the theme aimed at his or her age/maturity level.

The Apple II prices start at $1,295 and head upward, depending on the amount of memory needed or purchased. There are so many options available that your best bet is to get the rather thorough brochure from Apple before going down to your local computer store for hands-on experience. From our own observations, it's quite likely many of you will know more about the Apple II from the brochure than the staff at the store. This is because there are so many levels of features and potential usage that you and the store's staff might not be talking on the same level. ■

Phone the Future

(Continued from page 45)

which is less than one-third the time of the 110 baud stop bits.)

Baud rates can be intermixed for storage. Assume you have a Southwest Technical Products MC6800 personal computer such as the one E/E will use for our personal computing series. You might type in your program using a 110 baud model 33 teletype. Allow for corrections, proofing and changes and assume you have worked one hour loading the program. The final program—error free — if punched out on paper tape through the TTY might take 10 or 15 minutes to load it back in the next time you want to use it. But you can have a "hobby standard" 300 baud recorder connected to the computer, and instead of dumping the program out to the TTY you dump to the recorder at three times the TTY speed. The recording will take only some 3 or 4 minutes to save and reload into the computer. Some personal computing hobby recorders run 1200 baud, and that 10 or 15 minute program can be saved and loaded in a few seconds. (There are even higher baud rate recorders.)

Summing Up. Your first requirement for both personal and timeshare computers is the terminal. With the personal computing marketplace growing at a rate exceeding Jack's beanstalk, there are more and more ads from more and more companies offering surplus TTYs and CRT terminals, both kit and wired. Like the early days of anything, many won't be around tomorrow when you need service or parts, and some of what will be sold won't work too well. In selecting a terminal, stick to the well-known brands from companies that have deserved reputations. For example, you can always get parts and service for a model 33 TTY at reasonable prices. This is not necessarily true of any other TTY. If you're purchasing a used model 33 (which is about the only way to get one, because the waiting list for a new one direct from Teletype is months and months) make certain you get it from a company specializing in rebuilt models, such as National Teletype.

If you opt for a CRT terminal, the proven one is the ACT-1. If you want to go the build-it-yourself route, Southwest Technical Products is probably your best bet at this time.

For modems, Omnitec is considered by many the industry's workhorse. A used model 701A is available for about $150 and handles both TTY and RS-232 inputs. They have newer TTY models available in both semi-kit and wired versions for prices between $100 and $160.

The age of personal computing is upon us so you might as well climb aboard at the beginning. ■

DATAC 1000T

(Continued from page 43)

a Memory. The latter contains the frequency information for a selected set of words that the classifier is attempting to detect. Clearly the Analyzer, Classifier, and Memory must all be designed to function together since one depends on the other. Some classifiers have a training algorithm built into them that allows the user to repeat the word several times in order to home in on the right decision. The user, in this case, has a set of buttons to press to tell the computer the word he is trying to get across. So he presses the button for "stop baggage," for example, and then repeats the phrase "stop baggage" ten times so the Classifier is able to make a decision for that user's voice.

The future in this voice-controlled computer dreamland may bring us replacements for dictation services (just talk in the manuscript, the computer will do the typing job), voice-controlled security systems, or a television set that turns to the channel you whisper. Microcompressors can serve this dreamland as they become faster and more powerful. The future is almost here. ■

At The Mike

(Continued from page 39)

unplugged and updated with another one from SBE. The Nitron has other instructions built into its memory, such as automatically handling channels for ship-to-ship and ship-to-shore communications, and a memory for a channel you may want to save for later use. Accurate frequency control is another advantage of such a radio. The radio costs about $650. A new unit to be available very soon is called the Sea Command. In addition to the features of the Key/Com Fifty Five, this new marine radio will automatically scan through the 78 channels.

CB Radio. As with marine radios, a processor chip does the job of selecting channels, finding a conversation, quickly targeting on emergencies, and locking onto frequencies with great accuracy. The Key/Com 1000 is a mobile AM unit and was the first CB rig with a microprocessor and a keyboard. Giving the units commands to scan up and down, fast or slow, is like using a pocket calculator—you just have to know what key to press. The front uses a big, bright, LED display for showing all channels, power output, et cetera. There are two very nice ideas programmed into the Key/Com microprocessor. First, you can put up to ten of your favorite channels into a special memory, then hit a "go" button. The CB rig will scan those channels repeatedly until a conversation is found, at which point it stops. The processor is programmed to sit on each channel for one half second, then proceed to the next. Second, one channel can be placed in a special register and treated as a high priority channel. The processor will sample that channel every three seconds for activity, and, if incoming conversation is found, it will automatically switch the radio from the channel you are listening to, onto the high priority channel. A noise blanker, power output meter and signal strength meter are all part of the Key/Com 1000, which sells for about $295. ■

Chessboard Rumble

(Continued from page 50)

CHESS CHAMPIONS AT A GLANCE

	Manufacturer	List Price
BORIS	APPLIED CONCEPTS BOX 2582 GARLAND, TEXAS 75041	299.95
CHESS CHALLENGER	FIDELITY ELECTRONICS 5245 W. DIVERSEY CHICAGO, ILL. 60639	199.95
COMPU CHESS	STAID, INC. BOX 65 LARGO, FL. 33540	169.95

challenging game to the weekend player. As for the future—who knows? Perhaps even Bobby Fisher may someday hear a computer generated call of "checkmate!" ■

Space

(Continued from page 31)

most interesting features of GALAXY II, and is certainly a realistic simulation of economics. Depending on a planet's environment level, and level of development, it has what is termed Indexes of Efficiency and of Capacity. Simply put, the Index of Capacity governs how fast your population can grow and how quickly your planetary colonies can expand.

A player, who is considered his own race's leader, can invest in either of these indexes. This means you can decide, "Well, maybe I won't build that lovely fleet of Destroyer-class ships this turn. If I take the money and invest it in research (i.e. in my Index of Efficiency) then perhaps on the next turn I'll be able to build twice as many ships at only half the cost."

The interesting thing is, you can invest money (UUs—Universal Units) that you have or that you *don't* have. The latter method is known as deficit financing and is done by raising the income taxes of your colony to wartime levels. This is great, but does have certain drawbacks. For instance, after two turns of this, and if your investment has not shown a profit, your own colonists are likely to rebel against you. You can find yourself fighting not only a galactic war, but a few civil ones at the same time!

GALAXY II, which is play-by-mail with monthly turns, allows a person who is interested in computer gaming but who doesn't own a microcomputer to participate in a computer-run simulation.

The cost per turn is based on how many orders a Leader sends to his galactic minions. An average turn should cost in the neighborhood of five to eight dollars, no more than a magazine subscription. Brett Tondreau has a special introductory package available for twenty-five cents to any interested parties who write to: Brett Tondreau, GALAXY II, 20121-5 Leadwell St., Canoga Park, California 91306. ■

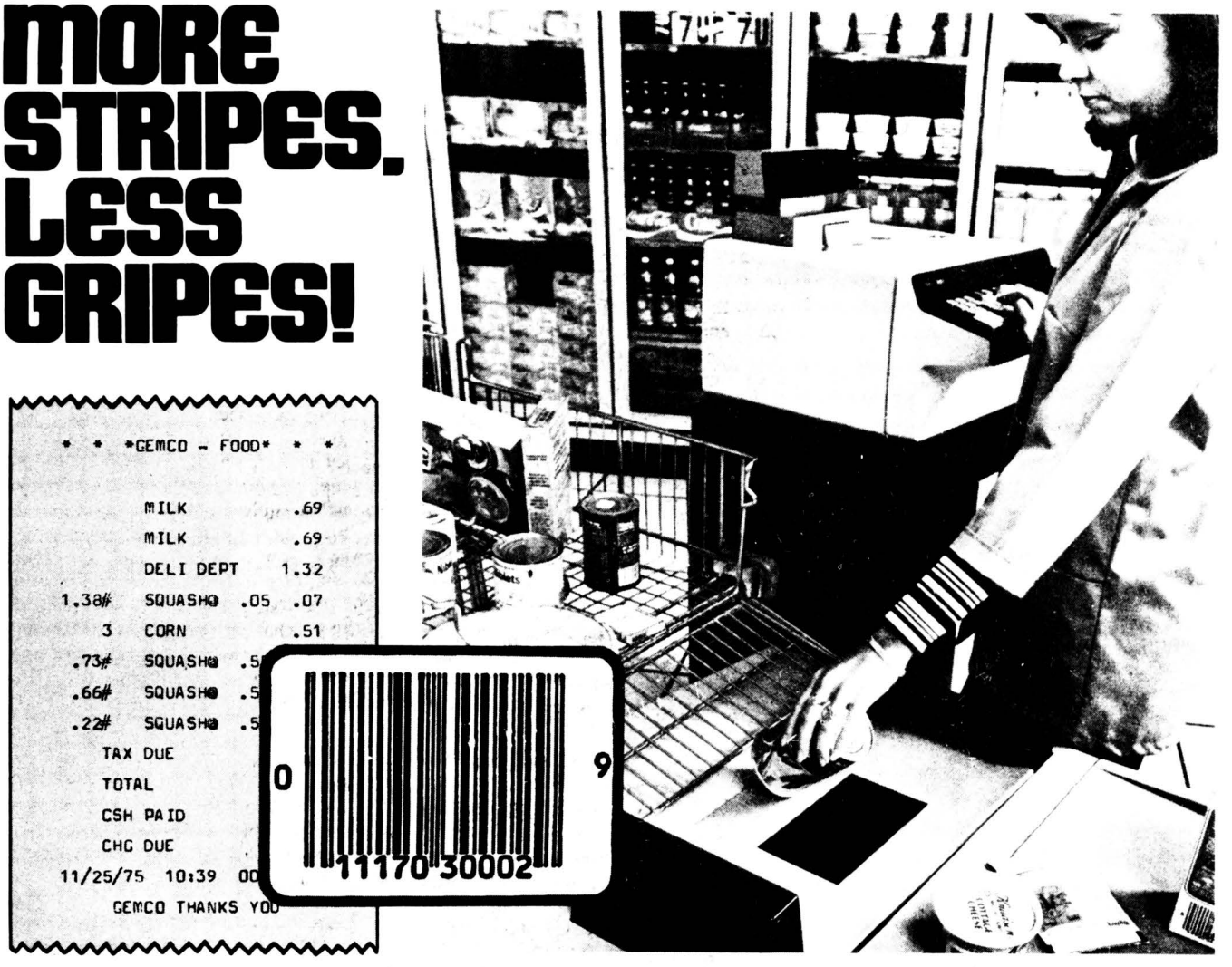
Hobby Computers

(Continued from page 7)

Although it's easy to get into hobby computing, it's not so easy to select *the right equipment for you* the first time out, and unfortunately, mistakes don't come cheap. In this issue of the HOBBY COMPUTER HANDBOOK we help you avoid expensive pitfalls by guiding you through the maze of computer hardware to equipment that offers the most performance and convenience at the lowest possible cost. We even give you some programs that have double value: Firstly, they are programs specifically tailored for the electronic and computer hobbyist. Secondly, they are instructional in that they illustrate different practical uses for the computer and the "tricks of the trade" that make the BASIC languages available to hobbyists even more powerful than they appear on the surface.

In short, this issue of the HOBBY COMPUTER HANDBOOK is meant as a desk reference for computer hobbyists. The more you read it, the more you'll get out of the features. ■

MORE STRIPES. LESS GRIPES!



Pricing Computers come to the local supermarket

by Shane Piroli

☐ When next you hand your favorite supermarket cashieress a tin of tuna don't be surprised if, rather than simply ringing it up, she whizzes it over a glass plate set into the top of the checkout counter and then presents you not with a simple register tape but with a completely itemized bill. Yes, computers have come to the local market!

Already, dozens of stores have installed the Universal Product Code System from IBM and it looks as if this is one wave of the future destined sooner or later to engulf us all. What will it all mean? Hopefully, lower prices and faster service at the consumer level and better inventory control and reordering procedures for the store manager.

It's all accomplished by a microprocessor computer which uses an optical scanner to "read" those black and white stripes which, by now, everyone and their Aunt Tillie has already noticed on nearly every package, box or can from even non-computerized stores. The fact of the matter is, manufacturers fully expect that all stores of large enough volume will become computerized, so they've already tooled up to label their products in computer-communicable language.

The optical scanner is beneath a glass-topped slot in the countertop (the slot being about a foot wide and several inches long). The unit can read the stripe-code no matter where it appears on the face of the glass. It can, if fact, read both "forwards" and backwards" so that it makes no difference which direction the product is moved along the scanners screen. If the cashieress actually misses the oversize screen no code will be read and the computer will thus inform its fallable human to try it again.

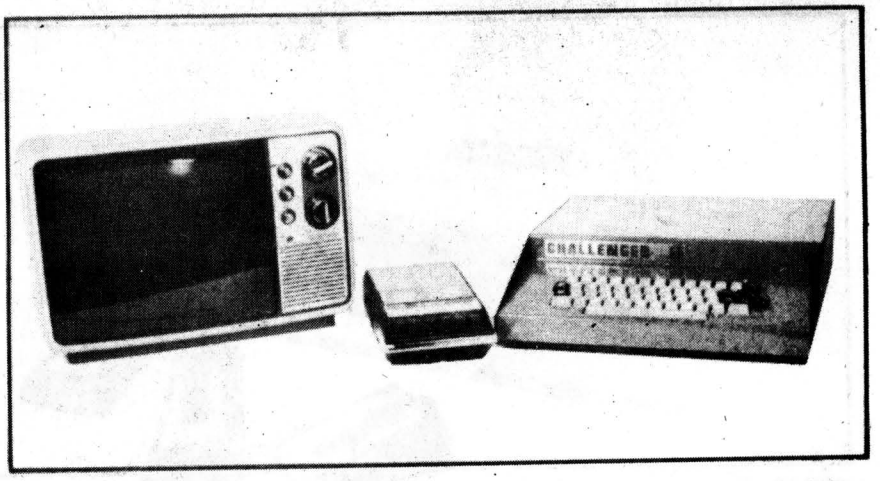
Products will, of course, still be labeled with normal prices via shelf-labels so that the consumer will be able to comparison shop and also know when he's reached the end of his wallet. Produce items, too fragile to be UPD stripe-encoded, will be priced normally and also marked with a code number the register operator can feed manually into the computer.

The computer, being a computer, is quite good at arithmetic and is capable of automatically computing sales tax, coupons, food stamps and both the amount given the cashieress and the change due the customer. Even better, all of this information is printed out on the register receipt and fully itemized; the consumer has a "hard" copy of all the fact which went into computing his bill so he can double-check it all at his leisure.

The even more futuristic possibilities inherent in this development are imagination staggering. Imagine going into a supermarket with no more money than what's in your bank account, and that not on you.

The computer age is still just beginning. Even now our lives are changing at every level in response to high technology moving into ever more and more areas of our lives. Such things as this supermarket computer seem to be one of the more positive results ■

COMPUTER



The Challenger IIP, new from Ohio Scientific, is a personal computer with 4K of RAM, BASIC in ROM, and a captive keyboard. The CPU is based on the new 6502 chip. Add a RF modulator or connect directly to a CRT for I/O display.

A—There is no such animal. Firstly, if your child is ready to enter High School, or already there, the school probably has an introductory course in either Data Processing or Computer Math. If your child wants to get into the design end, and has shown previous interest and ability in electronics, and really wants to play and experiment with the electrons, the Heathkit 6800 trainer is probably the only kit of practical value. (On a college level it's a whole different game.)

Q—*Which of the computer kit CPUs do you recommend for a beginner: 8080, 6800, 6502, or whatever?*

A—Tough question. The 8080 and its relative, the Z-80, often use the S-100 bus, for which there are many accessories. Unfortunately, the I/Os (inputs/outputs) are under software control and it can get somewhat expensive if you need ports for several peripherals. The 6800 I/Os are memory-mapped and it's cheap and easy to add peripherals. For example, the SWTP 6800 computer can handle up to ten peripherals and you simply purchase an inexpensive I/O card whenever you add another printer, terminal, recorder, etc. Also, the 6800 system allows a good intermix of serial and parallel I/O ports. You can do the same with the other CPUs but at much higher cost. Unfortunately, there are more "gadgets" for the S-100 bus than for any other bus or system, so you'll have to plan ahead.

Some personal computers using the 6502 are extremely powerful, but at the time this article was prepared there was little in the way of I/O equipment or even ports for peripherals.

Q—*What is meant by an integer BASIC?*

A—It means it cannot handle decimals. For example, the statement "PRINT 2/4" would return an answer of "00" instead of "0.5". Similarly, "PRINT 5/2" would return "2" instead of "2.5". Integer BASICS can be powerful in terms of graphics, etc., but they are useless for any school work involving even simple arithmetic. One exception to our rule of "No integer BASICs" is the Apple II computer, whose 4K resident integer BASIC is used to load Apple's notably good 16K BASIC.

Q—*How much memory would I need for a computer kit?*

A—12K will handle most BASIC interpreters running on an 8 bit system (8080, 6800, Z-80, etc.), though Apple requires 16K for their BASIC. The Heathkit LSI-11 system is a 16 bit system so you get the same results with half the memory: we would suggest at least 8K on the "Big Heath." Few computer kits come with enough memory to handle BASIC, or even an editor/assembler, so be sure to add in the cost of extra memory to the basic kit price.

Q—*What is meant by an "ASR Terminal"?*

A—ASR means **Automatic Send/Receive** and refers to a paper tape reader and punch accessory mounted as part of a teletype terminal. A used ASR TTY—the model 33—sells for about $900. Without the reader and punch you can get one for about $500 to $600; so if you have no need for paper tape you can save a bundle by getting a TTY without ASR. Just the printer part of the TTY, known as an RO33 for **Read Only,** sells for under $300, less than the cost of most "computer printer peripherals."

Q—*What is meant by "Hardware," "Software," and "Firmware"?*

A—**Hardware** *is* any equipment; computer, printer, even an individual integrated circuit. **Software** means a program, or instructions for the computer. **Firmware** is some form of program or instruction-set already in the computer, usually called a *resident monitor,* that makes it easy for the user to enter, or write programs. Firmware is ready as soon as power is applied.

Radio Shack's TRS-80 is a self-contained computer, with captive keyboard, based on the Z-80. The smallest version has 4K of RAM and an enhanced Tiny BASIC in ROM, and is complete with video monitor.